INTRODUCTION TO HTML & CSS

```
<p style="color:blue;">My page</p>
```

My page

CNIT Department

City College SF

TITLE & COPYRIGHT

Introduction to HTML & CSS

By CNIT Department Faculty of City College of San Francisco

Smashwords Edition

4th Edition

PREFACE

The objective of this e-book is to introduce readers to HTML and CSS. The files mentioned in this e-book can be found http://www.cdasilva.info/ebookfiles.zip. You can download the files and use them with the exercises proposed by this e-book.
After reading this e-book and following the exercises, you should be able to:

- Understand the history and evolution of the HTML markup language
- Write the HTML for a basic web page using simple text editors using
  - Headings
  - Paragraphs
  - Images
  - Lists
  - Table
  - Hyperlinks
- Use basic CSS properties to format color, style, and alignment of web page content
- Basic use of the W3C Validator to validate the HTML code
- Open an HTML document in the browser

This book was created by instructors of the CNIT Department of City College of SF in order to help students find a more economic textbook to be introduced to the basics of HTML and CSS as most textbooks in the market cover more areas and details then the introductory course designed for the CNIT curriculum.

The errata for this book can be found at http://www.cdasilva.info/errata.htm

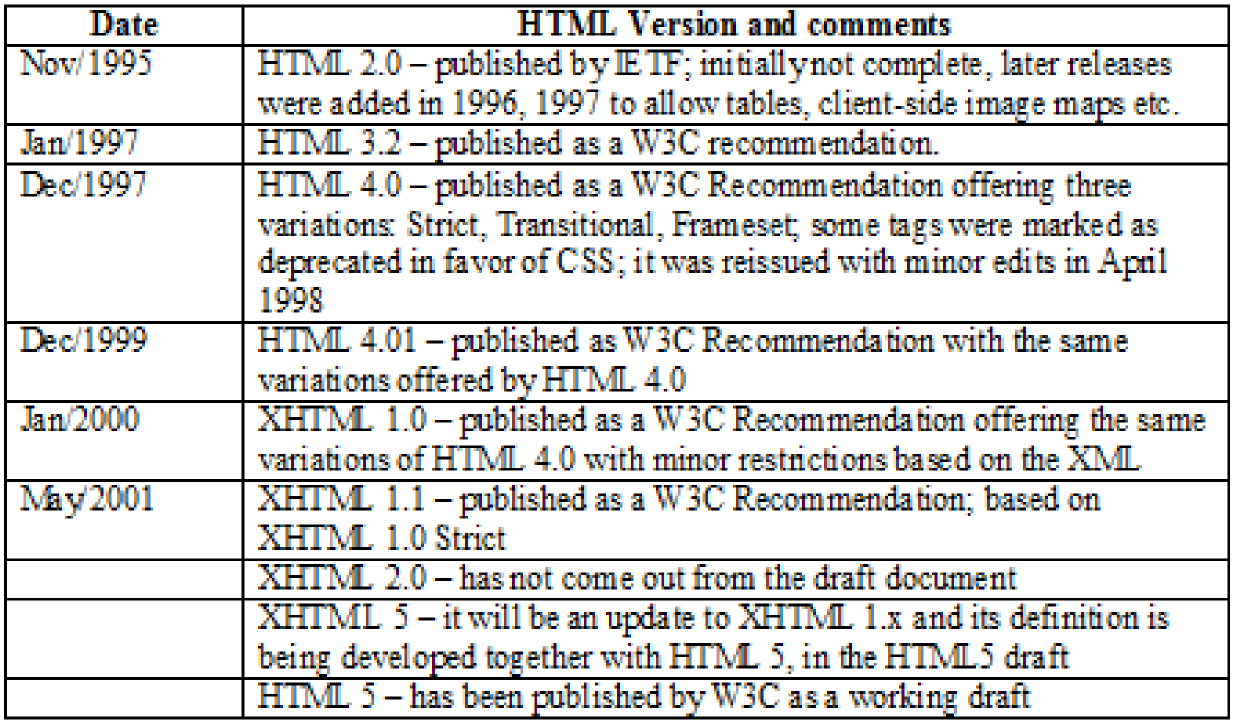
Introduction to HTML

What exactly is HTML?

HTML is not a programming language; it's short for Hyper Text Markup Language.

It is a nonproprietary language interpreted by the web browser. It came from SGML which was more complex, but was well known by Tim Berners-Lee that then created the HTML with the intention of sharing research papers over the internet.

In the table below, you will see the timeline for HTML:



Date	HTML Version and comments
Nov/1995	HTML 2.0 – published by IETF; initially not complete, later releases were added in 1996, 1997 to allow tables, client-side image maps etc.
Jan/1997	HTML 3.2 – published as a W3C recommendation.
Dec/1997	HTML 4.0 – published as a W3C Recommendation offering three variations: Strict, Transitional, Frameset; some tags were marked as deprecated in favor of CSS; it was reissued with minor edits in April 1998
Dec/1999	HTML 4.01 – published as W3C Recommendation with the same variations offered by HTML 4.0
Jan/2000	XHTML 1.0 – published as a W3C Recommendation offering the same variations of HTML 4.0 with minor restrictions based on the XML
May/2001	XHTML 1.1 – published as a W3C Recommendation; based on XHTML 1.0 Strict
	XHTML 2.0 – has not come out from the draft document
	XHTML 5 – it will be an update to XHTML 1.x and its definition is being developed together with HTML 5, in the HTML5 draft
	HTML 5 – has been published by W3C as a working draft

After HTML5, we had the release of HTML 5.1 in April/2013 (added the main element and removed the hgroup element) but the HTML 5.1 was only accepted as a W3C standard in November/2016!

Notice that HTML 3.2 is the first release from W3C but, who is W3C?

W3C stands for World Wide Web Consortium and it is a non-profit organization that creates and manges standards and guidelines with the objective of ensuring the long-term growth of the World Wide Web (WWW). The URL of their website is http://www.w3.org and the President of this organization is Tim Berners-Lee.

What is HTML used for?

Imagine you will write articles for the different sections of a newspaper. One of the first things you would do is gather all the news you want to publish. Then you would try to arrange the news in a structure that would attract the readers as well as make it easy for them to find the articles they want to read. If you look at any newspaper from that perspective, then you will easily understand the use of HTML.

HTML will be used to insert the content – the many words, sentences, paragraphs – that will constitute your web pages that together will make your website.

- In newspapers, you will be using special formatting for the headlines, right? In an HTML document, you will be mostly using the headings.
- In newspapers, you will be structuring the articles in paragraphs and you will do the same in an HTML document.
- In some articles of newspapers, you may have some information that you might need to structure in lists or tables which, sometimes, will be also necessary in your HTML document.

What tool should you use to write an HTML document?

There are many different tools in the market that can be used to write an HTML document (a web page) such as Adobe Dreamweaver, Microsoft Expression Web – these two are not free software but are tools very well known and used in the market. You will also find some free HTML editors that can be used as well and they are not being cited here as every year a new one comes up as freeware or shareware.

You need to understand that HTML is written in plain text and then the file is saved with the extension .htm or .html which browsers will then recognize as web pages and interpret as so. Being plain text, you can then even use simple tools as Notepad (for MS Windows users) or Text Edit (for Mac users).
If you are a Mac user, you should change some settings in the Preferences of the Text Edit in order to make sure that your files will be saved in plain text and NOT in rtf (Rich Text Format) which is generally the default for Text Edit.
You can also use Adobe Brackets that you can download from this website - http://brackets.io/ - it's a free web authoring tool that works perfectly fine for Mac and Windows users. You can also install some Extensions to Brackets, such as the W3C Validator, which allows you to validate your code from within the editor.
There are other editors such as Sublime Text that work equally for Mac and Windows; there is also Notepad++ and many other editors as I mentioned before.

Organizing your computer

The assumption for this course is that you know how to create, copy & paste, and delete files and folders in your computer – no matter if it’s a Mac or Windows computer. The other assumption is that you know how to install software in your computer to use them.

You should first create, preferably on the Desktop of your computer, a folder called **webroot** – this is just a way to make sure that we will be all saving the files in the same place. So, go ahead and create, on the Desktop of your computer, a folder called **webroot**.

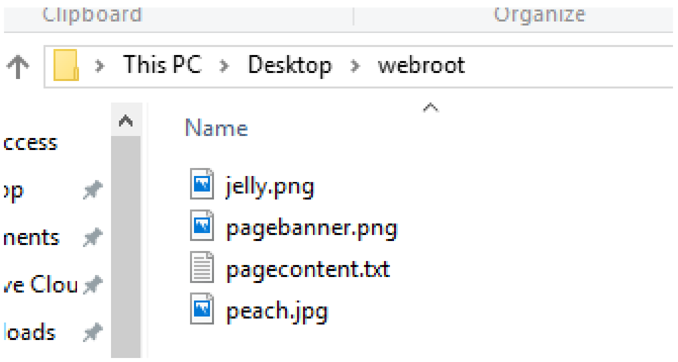
From this point on, every time I refer to working folder in our computer, I will be referring to the **webroot** folder you have created, ok?

In the Mac, you can generally use the File / New Folder in the menu to create a new folder. In Windows you can right-click on the place where you want to create the new folder and select New and then Folder. In both Operating Systems (Mac or Windows), when creating a new folder, you will need to type the name of the new folder (in our case, you will type **webroot**).

Make sure you have installed Adobe Brackets in your computer as well to use as your web editor.

Make sure you have access to some browsers options – Internet Explorer (Edge is how it’s called for Windows 10), or Firefox, or Safari, or Chrome (Google Chrome will be the browser that we will be referencing in this course but you can use any other browser you prefer).

After creating the **webroot** folder, copy all the files, that will be used by the tutorial, to that **webroot** folder – you can find the files at http://www.cdasilva.info/ebookfiles.zip (when you use this URL in the browser, you will not see a web page, you will see a file being downloaded to your computer and you need to remember that you will be downloading a zipped file called **ebookfiles.zip** and after you download that zipped file, you need to unzip it and copy the content of the folder **ebookfiles** to the **webroot** folder you created – ONLY COPY THE CONTENT OF THE FOLDER, NOT THE FOLDER). After you finish copying the files, you should have the following structure in your computer:



jelly.png
pagebanner.png
pagecontent.txt
peach.jpg

See that I have the **webroot** folder in my Desktop and inside the **webroot** folder I copied the files that were downloaded from the URL given in the last paragraph – those are the exact files you will find: jelly.png, pagebanner.png, pagecontent.txt, and peach.jpg.

You will now start writing HTML

Now that you have compared writing and organizing a newspaper with writing a website and that you know which tools can be used to write an HTML document, you are ready to start talking about some of the terms used in the "HTML world".

You will open your text editor tool – preferably for this course, we will be using Adobe Brackets (but you can use any other web editor / web authoring tool).

A break here **IF YOU ARE USING TEXT EDIT IN A MAC COMPUTER REMEMBER THAT YOU WILL NEED TO GO THROUGH SOME STEPS TO MAKE SURE THAT YOU WILL BE SAVING YOUR FILES AS PLAIN TEXT FORMAT** not rich text format – rtf – (which is generally the default in Text Edit). The settings change from version to version of Text Edit (version to version of the Mac OS) and you would need to research (using Google search engine for example – http://www.google.com) what are the adjustments you need to do in your Text Edit. In general, the adjustments would be something as you see listed below:

1. Open Text Edit in your Mac
2. Choose **Preferences** from the Text Edit application menu
3. Click the Plain Text radio button for New Document Format
4. Under **Saving**, click the check box to turn off “**Append ‘.txt’ extension to plain text files**”
5. Under **Rich Text Processing**, click the check box to turn on “**Ignore rich text commands in HTML files**”
6. Be sure that the check box for “**Wrap to Page**” is not selected

You should prefer a real web editor such as: Adobe Brackets, or Sublime, or Notepad++, etc. AVOID USING Notepad (in Windows) or Text Edit (in Mac) as they are really not good tools for web developers!

In Adobe Brackets, you will start with a blank new file and you do that by selecting, from the menu, *File / New*. Let’s then immediately save the file with the .htm or .html extension. You will save this first file as **myfirst.html**.
When using Adobe Brackets, you can select, from the menu, *File / Save As* and then make sure that you select the **webroot** folder and then type, inside the box called **File Name**, the name **myfirst.html**.
You will notice, as you work with Adobe Brackets, that this web editor uses some colors for the font to help you visualize the different parts of the code as you progress writing your code.

How do we start an HTML code from scratch? What should we write as the first line of an HTML document? If you open the source code of any modern website (web page), for example, Google, and you right-click on a blank space of the page and select the option View Source (if you are using Google Chrome browser), you will notice that the first tag is the one shown below:

```
    <!DOCTYPE html>
```

This is the DOCTYPE tag. W3C standards require this tag to be declared. This tag tells the browser which version of HTML is being used. In this case, the DOCTYPE tag is declaring that the HTML version being used is the HTML5. When you use the DOCTYPE tag, you will be able to validate your HTML code and the browser renders the page in standards mode instead of quirks mode.

What is Quirks Mode?
When CSS was still in its infancy, most browsers were not able to implement the CSS properly and used their own private layout algorithms. Currently CSS can be interpreted well by the browsers but there are still some web developers that do not use CSS appropriately and that is the reason browsers still make available that old mode called quirks mode.

Now, you will type the DOCTYPE tag in your blank document in the text editor you opened.

ATTENTION!!! NOTHING should come before the DOCTYPE tag! NOTHING, not even a comment!!!

The next tag is the html tag that starts the html document. So your code, with the html tag would be:

```
<!DOCTYPE>
<html lang="en">
```

Big Note!!! Very Important!!!

Sometimes, when you copy and paste code from slides, or Word documents, or pdf documents, the double quotes will be curved or inclined in your web editor (in Adobe Brackets) and this is because the format you see in those documents is rich text format and everything in HTML or CSS code needs to be in plain text. So, if you decide to copy and paste code and you notice that the double quotes (or single quotes) are inclined or curved, you just need to delete and retype the double quotes (or single quotes) in the web editor (Adobe Brackets) you are using. **SO, PAY ATTENTION TO THE DOUBLE OR SINGLE QUOTES WHEN YOU COPY THE CODE FROM A RICH-TEXT-FORMAT DOCUMENT!!!**

You will type the html tag in your **myfirst.html** right below the DOCTYPE tag as shown above. You will press ENTER (or Command in your Mac) twice and enter the closing html tag as shown below here:

```
<!DOCTYPE>
<html lang="en">

</html>
```

It’s advisable to have the opening html code like we coded (with the lang attribute) for accessibility reasons – we will talk more about accessibility when we learn about other tags.

The lang is called an attribute and, in this case, it’s an attribute of the html tag (html element). In the example above, we are stating that we will be writing our web page in English. There are other language code that can be used and this attribute is used by multilingual screen readers (a software that some users need to use, especially users with vision disability).

The two main sections of any HTML document

After the opening html tag, you will create the two main sections of any HTML document: the head and the body sections.

The head section will contain important information related to the web document but the content within this section WILL NOT be shown on the browser window with the exception of one tag that will be studied later on in this tutorial. Every content to be shown on the browser window needs to be within the body section.

In your myfirst.html file, type, right after the opening html tag, the opening and closing head and body tags as shown below:

```
<!DOCTYPE>
<html lang="en">
<head>

</head>

<body>

</body>
</html>
```

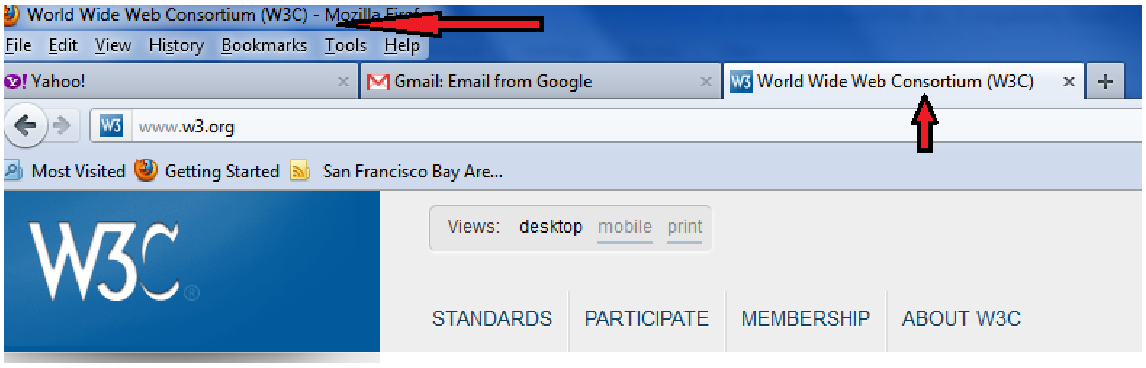
Notice that DOCTYPE tag is ALWAYS the first tag and the closing </html> is ALWAYS the last one. This means that NOTHING can come before the DOCTYPE tag and NOTHING can come after the </html> (the html closing tag). NOTHING!!!!!!!!

THIS IS THE BASIC STRUCTURE FOR EVERY SINGLE WEB PAGE YOU WILL CODE!!!

The head section

Between the <head> and </head> tags, you will be using important tags that will give more information about your web page to the browser.

The title tag should contain the title for your web page. Whatever you write as the title of your page will be shown in the top left of the browser when your page is opened or on the tab that your page is opened when many pages are opened in your browser. See the image below where the two red arrows show where the content of the title will be shown on the browser (the image shows a piece of the Firefox browser with 3 tabs opened).


World Wide Web Consortium (W3C) - Mozilla Firef
File Edit View History Bookmarks Tools Help
Yahoo!
Gmail: Email from Google
World Wide Web Consortium (W3C)
www.w3.org
Most Visited
Getting Started
San Francisco Bay Are...
W3C
Views: desktop mobile print
STANDARDS
PARTICIPATE
MEMBERSHIP
ABOUT W3C

You will write, below the <head> tag, the title tag as shown below:

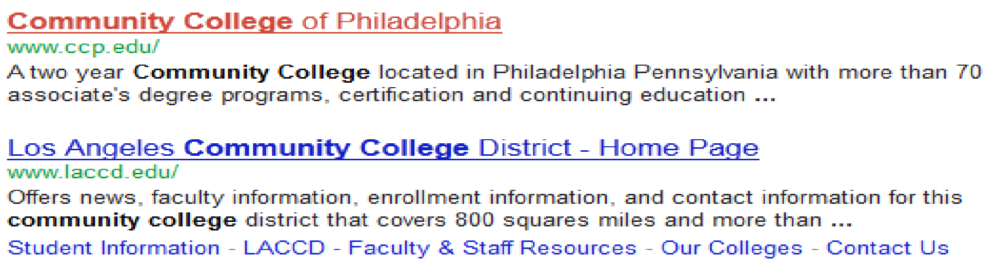
```
<!DOCTYPE>
<html lang="en">
<head>
<title>My first web page in HTML5</title>
</head>
<body>

</body>
```

```
</html>
```

Note:
Is it necessary to write each tag in a different line? What about the indentation shown for the title tag, is it necessary?
No and No! You could write your entire HTML code in one single line with no spaces between the tags but this method would make it more difficult to maintain the code in the future. But, keep in your mind that the browser will not consider the blank lines and the blank spaces you insert in your plain text file (your HTML code).

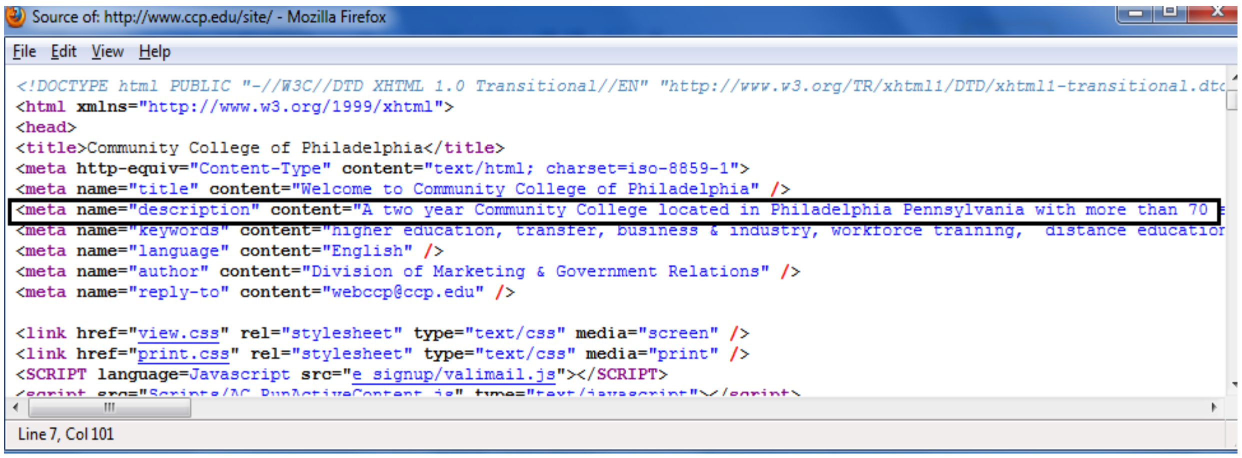
Another important tag used within the head section is the meta tag. There are many different types of meta tag but the ones you will learn in this tutorial and use in your myfirst.html page is the meta tag description and the charset.

The meta tag description is used to describe the web page. The content of this tag is used by some search engines when it lists the page in the search result. Observe the picture below that shows the search result list of a search engine and the image after that shows the part of the HTML code for the Community College of Philadelphia, where the meta tag description can be seen.



Community College of Philadelphia
www.ccp.edu/
A two year **Community College** located in Philadelphia Pennsylvania with more than 70 associate's degree programs, certification and continuing education ...

Los Angeles **Community College** District - Home Page
www.laccd.edu/
Offers news, faculty information, enrollment information, and contact information for this **community college** district that covers 800 squares miles and more than ...
Student Information - LACCD - Faculty & Staff Resources - Our Colleges - Contact Us



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Community College of Philadelphia</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="title" content="Welcome to Community College of Philadelphia" />
<meta name="description" content="A two year Community College located in Philadelphia Pennsylvania with more than 70
<meta name="keywords" content="higher education, transfer, business & industry, workforce training,  distance educatio
<meta name="language" content="English" />
<meta name="author" content="Division of Marketing & Government Relations" />
<meta name="reply-to" content="webccp@ccp.edu" />

<link href="view.css" rel="stylesheet" type="text/css" media="screen" />
<link href="print.css" rel="stylesheet" type="text/css" media="print" />
<SCRIPT language=Javascript src="e_signup/valimail.js"></SCRIPT>
<script src="Scripts/AC_RunActiveContent.js" type="text/javascript"></script>
```

Observe that the meta tag description contains the sentence that you see right below the link of the site for Community College of Philadelphia.

In your myfirst.html page, you will type the meta tag description right below the title tag and also the meta charset as seen below:

```
<!DOCTYPE>
<html lang="en">
<head>
<meta charset="utf-8">
<title>My first web page in HTML5</title>
<meta name="description" content="This is my first page using HTML5
and inline CSS">
</head>
<body>


</body>
</html>
```

Have you noticed that the meta tag does not have a corresponding closing tag? Have you also notice that you used other “words” such as *name* and *content* to complete the meta tag?
The tags you had used up to this point (html, head, body, title) are called two-sided tags as they have opening and

closing tags – example, the <title> and </title>. The meta tag is a one-sided tag and its closing would be represented by the forward slash (if you were using XHTML) or just the > to close the tag (for HTM5 – the one we are using) – it is not an error if you use the forward slash to close the one-sided tags in HTML5.

The two “words” you see in the meta tag (**name** and **content**) are called attributes (like the lang that was used as an attribute for the html opening tag) and they are required for some tags like, for example, the meta tag. Other tags will contain attributes to modify or add something different to the tag – you will see other examples later.

The meta tag you see before the <title> element is use to specify the character encoding of your content. An HTML document (page) can have only one charset and most of the web pages will use the UTF-8 character encoding. If you are more curious about this topic, you can check the W3C page https://www.w3.org/International/questions/qa-choosing-encodings The meta charset tag used to be longer such as `<meta http-equiv="content-type" content="text/html; charset=utf-8" />` but in HTML5, you just need to type `<meta charset="utf-8">` but notice that it would not be wrong to write the longer version of that meta tag but why would you do that if you can write a shorter version?

There are other tags that can be found in the head section:

- The script tag (**<script>…</script>**) – used to insert JavaScript code in the HTML document
- The link tag (**<link…>**) – used to link external files to the HTML document, for example, to link an external CSS file

Important Note!!!

The way we coded the meta charset coming even before the <title> tag is a recommendation from the W3C that it’s better to have the meta charset as the first tag in the <head>…</head> section of the page because this tag affects the character set for the entire document and needs then to be applied from the offset having then a higher priority than any other tag in the head section. It’s just a recommendation!!!

Inserting content within the body section

You will use the text found in **pagecontent.txt** file as the content of your web page. You should copy the whole content of this file and paste it after the body opening tag – all the text should be between the <body> and </body> tags. Your code will look like what you see below (some lines of the pagecontent.txt file were skipped to fit the space better). After pasting the content from pagecontent.txt into your plain text file, do not forget to keep saving your myfirst.html file!

```
<!DOCTYPE>
<html lang="en">
<head>
<meta charset="utf-8">
<title>My first web page in HTML5</title>
<meta name="description" content="This is my first page using HTML5
and inline CSS">
</head>
<body>
Cakes, Jelly, Desserts
Have you ever wonder why everything that is sweet tastes so great?
Because it's sweet!
...
...
415-639-2113
Copyright of Amazing Desserts (c) 2011
</body>
</html>
```

Working with HTML tags to organize content

After you save your **myfirst.html** file, if you open it in any browser, you will see the content with no formatting, no paragraphs or a single separation to make it readable. The sentences will come one after the other in the same line as the browser ignores completely any line break or indentation that you manually included in your web editor. In order to organize the content in the browser, you need to enter the HTML markup code.

Note:
How do you open your file (myfirst.html) in a browser? Remember that if you are following this tutorial, you saved your file inside the folder called webroot. Well, there are two ways you can open an HTML document (saved as .htm or .html) in a browser:
a) you can locate the file in your computer, right-click on it and choose Open with and select a browser that you want to open your file with
OR
b) you can open the browser you will be working with and then you can locate the file in your computer and then click once on it and drag it inside the opened browser window
OR
c) you can open the browser and then select, on the menu of the browser, the File / Open and then locate the .html or .htm file you want to open inside your computer.

HTML5 introduced some interesting tags to organize contents in “boxes” that will make sense in most cases.

Those tags are: header, section, article, aside, footer. These are all two-sided tags (remember the concept of two-sided tags? The opening and corresponding closing tags?).

As you can imagine, the header tag will be used to surround the content that would be part of a headline, or the top part of your web page. You will include the <header> tag just after the <body> tag and before the ***Cakes, Jelly, Desserts*** headline of your content. Then, after the word ***Desserts***, you will type the closing header tag </header>. That part of the code is being shown below – see the <header> and </header> tags in bold and red below:

```
<!DOCTYPE>
<html lang="en">
<head>
<meta charset="utf-8">
<title>My first web page in HTML5</title>
<meta name="description" content="This is my first page using HTML5
and inline CSS">
</head>
<body>
<header>Cakes, Jelly, Desserts</header>
Have you ever wonder why everything that is sweet tastes so great?
Because it's sweet!
...
...
415-639-2113
Copyright of Amazing Desserts (c) 2011
</body>
</html>
```

Note:

As developers get more comfortable with these new structural tags - <header>, <section>, <article>,

<footer>, <aside>, etc. – you will see more examples of pages that are coded with them but you may also see the <div> tag being used instead such as: <div id=”header”> or <div id=”heading”> being closed by </div> instead, for example, of the <header>…</header> tags.

The section box will start exactly after the closing header tag. After typing the opening and ending section tags, your code will look something like you see here and, again, due to space, some parts of the content were substituted by “…”.

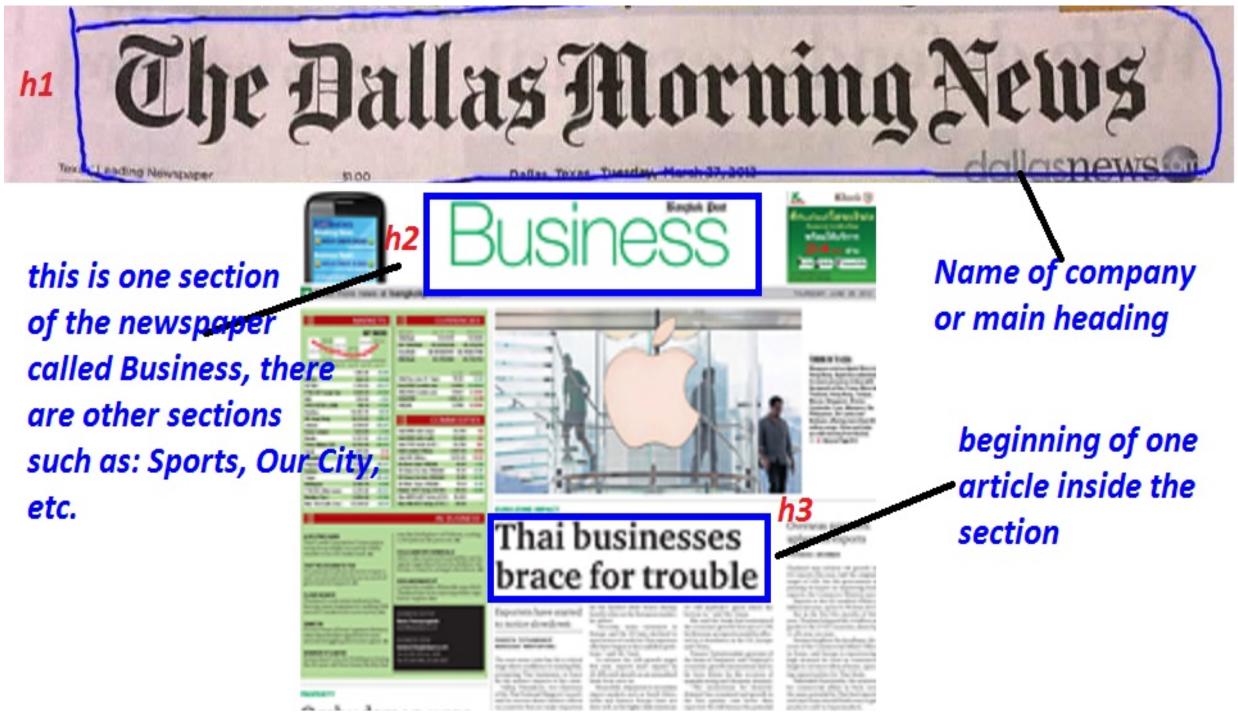
```
<!DOCTYPE>
<html lang="en">
<head>
<meta charset="utf-8">
<title>My first web page in HTML5</title>
<meta name="description" content="This is my first page using HTML5
and inline CSS">
</head>
<body>
<header>Cakes, Jelly, Desserts</header>
<section>
Have you ever wonder why ...
...
Cake – any type $2.50
</section>
Vocabulary of French desserts
...
415-639-2113
Copyright of Amazing Desserts (c) 2011
</body>
</html>
```

Within a section (between <section> and </section> tags), you can have one set or many sets of articles (<article>…</article>). Comparing again with the newspapers, imagine the Sports section of the papers that might have one single article about one sport event or this section might have many articles about different sport events.

Note:
The <section> and <article> tags still create a lot of confusion among web developers. Some will prefer to create sections inside a single article, some will prefer to do the way we are doing here.
What needs to be understood is that both these tags – and also the other ones: <header>, <footer>, <aside> - will not position and/or format the content by themselves. You still need to use CSS to do it.
Also, you need to understand that the idea behind those tags is to bring more semantics to the arena of web development and you are not forbidden to use more than one header, or footer tags within your HTML document.

Opening big parentheses here to talk about structure of web pages and semantics

Whenever we write the content of a web page, we always need to have in mind that most of the web page development standards come from the desktop publishing industry and, based on that, let’s take a look at the image below of a page of a newspaper:


h1
The Dallas Morning News
h2
Business
this is one section of the newspaper called Business, there are other sections such as: Sports, Our City, etc.
Name of company or main heading
h3
Thai businesses brace for trouble
beginning of one article inside the section

As you can see in the image above, the main headline of the page is the name of the newspaper "The Dallas Morning News" and that's the one with the biggest font size you can see (in our HTML "world", this would be the main heading of the web page, the h1 tag that we will see later). Then, you see the other heading "Business" which would be the heading of one of the sections of a newspaper (generally a newspaper is divided in sections, right? We can have the Business, the Sports, the World, etc.) and in the HTML "world", that heading would be a h2 tag. Then you see the smaller heading "Thai businesses brace for trouble" which shows the heading of one of the articles of the Business section and in the HTML "world" that would be a h3 tag.

As you can see, the main idea is to create a semantic structure making sure that you do not code a bigger heading after smaller headings.

END OF THE BIG PARENTHESES HERE

In this web page, you will consider the existence of two articles: The first will contain the short history about desserts up to the list of types of desserts; the second article will contain the table with the types and prices of desserts available at the shop.

With the information from the previous paragraph, you should then type the two sets of opening and closing article tags in your code and it will look as what is shown below (again, using “…” to omit some lines and due to space limit).

```
<!DOCTYPE>
<html lang="en">
<head>
<meta charset="utf-8">
<title>My first web page in HTML5</title>
<meta name="description" content="This is my first page using HTML5
and inline CSS">
</head>
<body>
<header><h1>Cakes, Jelly, Desserts</h1></header>
<section>
<h2>About our Business</h2>
<article>
<h3>How it all started</h3>
Have you ever wonder why ...
...
Others
</article>
<article>
<h3>About our products</h3>
Our products and prices
...
Cake – any type $2.50
</article>
</section>
Vocabulary of French desserts
...
415-639-2113
```

```
Copyright of Amazing Desserts (c) 2011
</body>
</html>
```

Note that we have also included heading (h2 and h3 tags) right after the opening section tag and then right after each opening article tag (you will see the tags in bold in the code above). The reason for that is to follow the recommendation from W3C that states that a section or an article should have a heading (h2, or h3, or h4, etc…). We also included a h1 tag around “Cakes, Jelly, Desserts” as, in our case, it identifies the most important information on the page – it’s recommended to have only one h1 per page (NOT PER WEBSITE – remember that a website can have one or more pages). **Headings should NEVER be used as a way to simply increase the size of the text – you will learn CSS to do that!**

Now that you have the section and articles of your web page ready, you will use the aside tag around the part of the content dedicated for vocabulary. The aside section should represent the content of the page that you might want to put aside (like the name of the tag). It might be some text that is not exactly part of the content of your page and might be used by customers just as a type of reference. In some web pages, you might see latest news, advertisement, interesting links, etc. in a separated session (“box”) of the web page. It can also represent any extra information you are providing to users, not necessarily displayed on the right or left side of the content, but that is definitely not part of the content. The opening aside tag will be placed before ***Vocabulary*** and the closing aside tag will be placed before ***Traditional cakes, jelly and desserts for you!***. The placement of the aside tags are shown below – it has been omitted here the upper part of the code (from the DOCTYPE tag up to the closing head </head> tag and that portion is represented by “…”):

```
     ...
<body>
<header><h1>Cakes, Jelly, Desserts</h1></header>
<section>
<h2>….</h2>
<article>
<h3>…</h3>
Have you ever wonder why ...
...
Others
</article>
<article>
<h3>…</h3>
Our products and prices
...
Cake – any type $2.50
</article>
</section>
<aside>
Vocabulary of French desserts
...
glace – ice cream
</aside>
Traditional cakes, jelly and desserts for you!
...
415-639-2113
Copyright of Amazing Desserts (c) 2011
</body>
</html>
```

The last "box" you will code will be the footer. The footer will contain information that, in general, you will find in the bottom part of a web page. In the case of this page, the footer will contain the information about the company and the copyright information as well. After entering the opening and closing footer tags, your code should look like what is shown below – here we are just showing the code from the opening aside tag and below just to show where to include the opening <footer> and the closing </footer> tags.

```
     ...
<aside>
Vocabulary of French desserts
...
```

```
glace – ice cream
</aside>
<footer>
Traditional cakes, jelly and desserts for you!
...
415-639-2113
Copyright of Amazing Desserts (c) 2011
</footer>
</body>
</html>
```

Note:

These new HTML5 tags, especially the header and footer, can be used more than once in a single page. This would not be considered an invalid code. These tags were created for semantic purposes only and they WILL NOT automatically place the content between these tags on the browser, on the right location, for you. The content will be placed on the browser from top to bottom as it is downloaded and parsed by the browser (as the HTML code is read and parsed by the browser) – this means that if a developer decides to put the footer at the top of the code, before the header, that's where it will be shown in the browser

Note:

The tags you have just used could be compared with the div tag that web developers have used and some web developers continue to use. For example, the header tag you have inserted in your page could become the identifier of a div tag as seen in the code below:

```
     ...
<body>
```

```
<div id="header"><h1>Cakes, Jelly, Desserts</h1></div>
     ...
</body>
</html>
```

Note:
The main difference is that in this case you are not using a tag to define the “box”. You are using the div tag with an identifier (id attribute) – remember the previous Note right after the use of the <header> tag! The value for id can be any that bring some meaning to the “box” (the element) you are creating. The value given to the id cannot be repeated in another tag WITHIN THE SAME HTML document (same page). The id value needs to be unique within the scope of the web page (HTML file). Another difference will come when you code the CSS to format and position that “box” inside the page.

Now that you have all the parts of the page separated by tags (marked up with tags), you need to start using other HTML code to continue to format the content of the page.

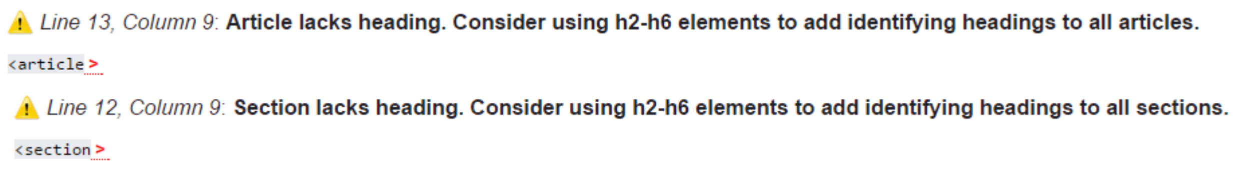
We have already used the h2 tag around ***Cakes, Jelly, Desserts***. The heading tags are used exactly like the headlines you see in any newspapers – the letters will be bolder and, for some of the heading tags, the letters will also be bigger. You have already seen an example of the use of these tags when we talked about the section and article tags, remember? These tags can go from h1 to h6. The h1 will present a bigger font with the h6 presenting the smallest one (depending on the resolution of the screen, it might be really hard to read the content of the h5 and h6 tags). The heading tags are two-sided tags (they have the opening and the closing tags).

After you enter the h1 opening and closing tag around ***Cakes, Jelly, Desserts***, your code will be like the one shown below. Let's also include the h2 tags to surround the words ***Our products and prices*** and ***Vocabulary of French desserts*** as shown here as well – only showing the parts before and after where the new tags are being included and the rest of the content being represented by "...".

```
     ...
<h2>Our products and prices</h2>
...
Cake – any type $2.50
</article>
</section>
<aside>
<h2>Vocabulary of French desserts</h2>
     ...
</body>
</html>
```

Note:
Do not forget to save your html file every time you change it! Now, you do not need to use the File / Save As from the menu of Adobe Brackets, you can simply use File / Save.

Here is a better explanation of why we included those h2 and h3 tags for the section and article tags respectively. W3C recommends that we create a heading for each section and article we include in our code. The heading has to be between h2 and h6 because supposedly we included an h1 somewhere else in the code! When validating the code we are creating, you may then receive a warning like shown in the picture below (of course, the line and column numbers might differ from your case when validating your code which we will be doing later):



```
Line 13, Column 9: Article lacks heading. Consider using h2-h6 elements to add identifying headings to all articles.
<article>
Line 12, Column 9: Section lacks heading. Consider using h2-h6 elements to add identifying headings to all sections.
<section>
```

If you want to fix these warnings, you can simply add, right after the <section> opening tag and right after the <article> opening tag an h2, or h3, or h4, or h5, or h6 – like we have done. Remember that big parentheses where we showed the image of a page of a newspaper explaining the main heading of the page and the headings of section(s) and/or article(s).

You need now to identify the paragraphs in your content using the paragraph two-sided tag (<p>…</p>). The first paragraph starts with ***Have you wonder***… and ends at ***Because it's sweet***. The other paragraphs can be easily recognized by the content of the text but you can modify it a little bit if you so desire. Besides the first pair of <p> and </p> tags proposed, to follow this exercise, you should insert opening paragraph tags <p> before the following:
Sweets were appreciated…
During the old times…
Many nice desserts…
Traditional cakes, jelly…

You should have closing paragraph tags </p> after the following:
…***and sugar was manufactured.***
…***enjoy it on special occasions.***
…***All Culinary Schools.***
…***Amazing Desserts (c) 2011***

The code below is just showing the code for the first and the last paragraphs mentioned above so you can see the placement of the opening and closing paragraph tags <p> </p>:

```
     ...
<body>
<header><h1>Cakes, Jelly, Desserts</h1></header>
<p>Have you ever wonder why ... Because it’s sweet!</p>
...
<p>Traditional cakes, jelly and desserts for you!
123 Baker Road, San Francisco, CA 94112
415-639-2113
Copyright of Amazing Desserts (c) 2011</p>
</body>
</html>
```

Remember to save the file! Now, when you open your **myfirst.html** file in the browser, you will notice that the paragraphs will be separated by a blank line and the content within each paragraph will be wrapped automatically. This spacing is provided by the browser (by the internal user agent stylesheet of the browser) and, of course, this spacing can all be changed by CSS.

Look at the last paragraph (located in the footer) – you do not want the paragraph written in a single line, right? But what if you do not want to have blank lines separating each of the lines in that paragraph? In this case, all you need is a line break and this is achieved with the HTML one-sided tag <br>.

Look at the code below to see how you will be adding the <br> tag in your web page:

```
     ...
<body>
<header><h1>Cakes, Jelly, Desserts</h1></header>
<p>Have you ever wonder why ... Because it’s sweet!</p>
...
<p>Traditional cakes, jelly and desserts for you!<br>
123 Baker Road, San Francisco, CA 94112<br>
```

```
415-639-2113<br>
Copyright of Amazing Desserts (c) 2011</p>
</body>
</html>
```

By now, you would probably like to insert some comments in your code so later it can help you remember what you learned in this tutorial. The way to insert comments in any HTML document is by using the tags `<!--` and `-->` to respectively start and end the comment lines. You can insert comments in the head section or in the body section of your document and as many comment lines or different comments you feel necessary. Take a look at the code below to see an example of comment lines that were inserted in the head section of **myfirst.html** document.

Some important details to remember: Comments WILL NOT be shown on the browser when you display your web page and **comments CANNOT be inserted before the DOCTYPE tag**.

```
<!DOCTYPE>
<html lang="en">
<head>
<meta charset="utf-8">
<title>My first web page in HTML5</title>
<!-- here is some comment -->
<meta name="description" content="This is my first page using HTML5
and inline CSS">
</head>
<body>
<header><h1>Cakes, Jelly, Desserts</h1></header>
<p>Have you ever wonder why ... Because it's sweet!</p>
...
<p>Traditional cakes, jelly and desserts for you!<br>
123 Baker Road, San Francisco, CA 94112<br>
415-639-2113<br>
Copyright of Amazing Desserts (c) 2011</p>
</body>
</html>
```

Note:
You need to make sure you use the comment tags correctly. You type `<!--` then you leave a blank space between the last dash (-) and then you start typing your comment. When you finish your comment, you leave another blank space before you type `-->`

The word creme is a French word that should be written as crème (observe the sign over letter e). In order to make this type of character show in the browser, you need to use the special characters code. There are many websites that show a list of special characters that a web developer can use when coding their pages – here are two websites that you can bookmark to have as a future reference for you: http://www.degraeve.com/reference/specialcharacters.php, http://www.ascii.cl/htmlcodes.htm (it has the hexadecimal representation as well), and http://webdesign.about.com/library/bl_htmlcodes.htm

As you can see in both websites mentioned in the previous paragraph, to have the letter e with the grave accent on it, you would have to code the following **&egrave;** (this is the friendly code) or **&#232;** (this is the numerical code) or **&#xE8;** (this is the hexadecimal code). Note that you must always have the ampersand (&) to start a special character in HTML followed by the friendly name and then the semicolon (;) or followed by the pound sign (#), then the numerical or hexadecimal code and the semicolon (;). So, a sentence such as `the best pátisserie`, would be coded as `the best p&aacute;tisserie` and if using the numerical version of the special character, it would be `p&#225;tisserie`. Notice that I continued to type the rest of the word right after the special character code finishes (right after the semicolon (;) that represents the end of the special character).

Observe the &egrave; in the code below in bold and red color – that's the way you should code in your myfirst.html file.

```
     ...
<body>
...
<h2>Vocabulary of French desserts</h2>
cr&egrave;me caramel – flan
...
</body>
</html>
```

If you have been curious enough while looking at the list of available special characters in the websites referred to you, you probably observed that we can also substitute the (c) found in our page by the special character that creates the copyright sign ©. How would you do that? This is another special character so, try doing some research and try doing it yourself before you look at the code below where you will find, in the footer part of the page, the copyright symbol special character being coded:

```
     ...
<body>
...
...
<p>Traditional cakes, jelly and desserts for you!<br>
123 Baker Road, San Francisco, CA 94112<br>
415-639-2113<br>
Copyright of Amazing Desserts &copy; 2011</p>
</body>
</html>
```

Just for you to see how important is to know how to use special characters, imagine if you wanted to write an HTML tutorial as a web page and you would need to write a sentence as:
“One of the most used HTML elements is the <p> element to create a paragraph”
You could imagine coding this paragraph as:

```
<p>One of the most used HTML elements is the ???????????</p>
```

Well, the ????? is to show a big problem we are facing here! If I write, in the HTML code, in the middle of that sentence, the <p> directly like this, the browser would not understand and would point this as an error saying that I cannot write one paragraph element inside another – this <p> in the middle would conflict with the <p> you wrote in the beginning, right? Even if you were trying to write a <br> or <h2>, it would create the same confusion as the browser would “parse” that text as HTML code and not as a simple text!

The solution would be exactly to use special characters and, in that case, I could easily write that content by simply having my code like this:

```
<p>One of the most used HTML elements is the &lt;p&gt; element to create a paragraph</p>
```

Now, everybody is happy! The browser will understand that you have a single paragraph with one sentence and then will not try to interpret another paragraph in the middle of the sentence, it will just interpret the special characters and convert them to < and > respectively.

Inserting a thematic break

What is a thematic break? A thematic break is when you have a break in the theme of your content or you may want simply to call the attention to a new part of the content of your web page. So, if you want to insert a visual separation in your web page, the easiest way is by using the horizontal rule tag <hr>. This is another one-sided tag that you may find coded as <hr />

So, let’s continue working on our **myfirst.html** file and include a thematic break (<hr> tag) between the two articles we have in our content. Your code will be like the one shown here:

```
     ...
<body>
<header><h1>Cakes, Jelly, Desserts</h1></header>
...
</article>
<hr>
<article>
...
</article>
...
</body>
</html>
```

Note:
Always save your file!!! Open it in the browser again and see how it looks now!
By the way, while working with your file, you can leave it opened in the browser and after you modify your code and save the html file, you can simply refresh the content of the browser and your web page will be reloaded with the modifications you have saved.

Tables – using them correctly

In many instances we need to make use of tables exactly to present tabular data. For example, in the web page we have been working on, the prices of the cakes can be considered tabular data and would be better presented in a table.

Some years ago, some web developers started using tables to lay out entire web pages, inserting all the content of the pages in cells of tables. It was an easy and straight-forward way to lay out the content of pages within different columns and lines. Recently studies have determined that this is not a good approach and CSS should be used instead due to download time but especially due to Search Engine Optimization (SEO) practices.

Here, we will learn how to code a table in HTML to present the prices of the cakes in a tabular way.

A table, in HTML, starts with the <table> tag and, of course, will end with the closing </table> tag. Each row of the table will be surrounded by the <tr> and </tr> tags and within each table row, you will find as many table cells you need and the content of the cells will be surrounded by the <td> and </td> tags.

So, let's continue working with myfirst.html file and insert the following code that will substitute all the lines from "**Ice cream w/ jelly**" up to **"$2.50**"

```
<table>
<tr>
<td>Ice cream w/ jelly</td>
<td>$1.70</td>
</tr>
<tr>
<td>Fruit mousse - small</td>
<td>$2.00</td>
</tr>
```

```
<tr>
<td>Cake – any type</td>
<td>$2.50</td>
</tr>
</table>
```

So, your code would look like what you see below:

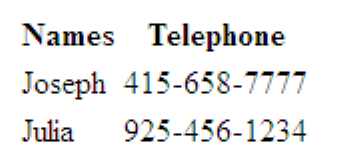
```
     ...
...
</article>
<hr>
<article>
<h2>Our products and prices</h2>
<table>
<tr>
<td>Ice cream w/ jelly</td>
<td>$1.70</td>
</tr>
<tr>
<td>Fruit mousse - small</td>
<td>$2.00</td>
</tr>
<tr>
<td>Cake – any type</td>
<td>$2.50</td>
</tr>
</table>
</article>
</section>
...
</body>
</html>
```

You might need to build a table that will have the first row being the header. This can be easily achieved in HTML if, within the first row (within the <tr> and </tr> tags) you include the headers of the columns within <th>…</th> tags (table heading tags) instead of the <td>…</td> tags we used. The code below:

```
<table>
<tr>
<th>Names</th>
```

```
<th>Telephone</th>
</tr>
<tr>
<td>Joseph</td>
<td>415-658-7777</td>
</tr>
<tr>
<td>Julia</td>
<td>925-456-1234</td>
</tr>
     </table>
```

…would present the following result when opened in a browser:



Names	Telephone
Joseph	415-658-7777
Julia	925-456-1234

Notice that “Names” and “Telephone” came centralized within the cell and in bold. That’s the result of using the <th>…</th> tags which means that, by default, the browser presents the content of th tags centralized within the cell and with bold font.

Note:

Take a look at your web page in the browser! Do you see the table with borders or without borders? It does not look so nice, right? You can make it look nicer using CSS. This will come later!

Lists

We find three different types of lists in HTML:

- Unordered list
- Ordered list
- Definition list

Unordered Lists: each item is shown, by default, with a bullet – those lists are generally used to present a list of items that not necessarily need to be shown in any specific order (such as the list shown here above to show the existing types of lists).

Ordered Lists: each item is shown, by default, with a number – those lists are generally used to present a list of items that need to be shown in a specific order – Example: steps to be followed to bake a chocolate cake.

Definition Lists: the items come in pairs – the term to be defined and the definition (description). This list is used for glossary, vocabulary, questions & answers, etc.

The unordered list is coded as:

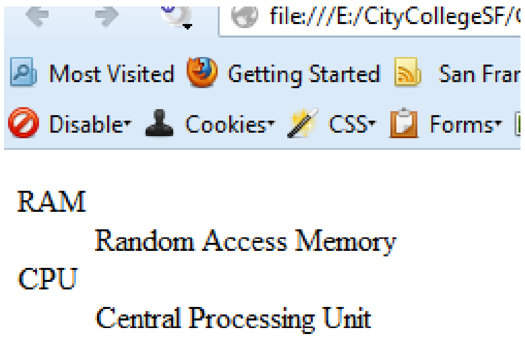
```
<ul>
   <li>item 1</li>
   <li>item 2</li>
</ul>
```

Of course, you can have more than two items as shown in the example above. Each item needs to be surrounded by the <li> and </li> tags.
The only difference between the unordered list and the ordered list is that the last one starts and ends with the <ol> and </ol> tags respectively. Each item in the ordered list is also surrounded by the <li> and </li> tags.

The definition list starts with the <dl> tag and ends with a closing </dl> tag. Between these two tags, you will have the pairs of terms and definitions. The terms are presented surrounded by the <dt> and </dt> tags and the definitions by the <dd> and </dd> tags. So, the following code

```
<dl>
   <dt>RAM</dt>
   <dd>Random Access Memory</dd>
   <dt>CPU</dt>
   <dd>Central Processing Unit</dd>
</dl>
```

Would be shown in the browser as



```
RAM
        Random Access Memory
CPU
        Central Processing Unit
```

Notice that the definition of the term is presented with some indentation in relation to the term that is being "defined". This is the default way that browsers present a definition list.

Let's include the unordered list code in our web page – **myfirst.html**. Open the file in your text editor and let's make the text shown below be presented as an unordered list:
Ice cream
Mousse
Cakes
Jelly
Others

Your code should look like:

```
     ...
<body>
...
<ul>
<li>Ice cream</li>
<li>Mousse</li>
<li>Cakes</li>
<li>Jelly</li>
<li>Others</li>
</ul>
...
</body>
</html>
```

Our page also brings the possibility of having a definition list. Let’s get back to our **myfirst.html** file and make the part related to Vocabulary become a definition list.

After inserting the definition list, your code should look like:

```
     ...
<body>
...
<h2>Vocabulary of French desserts</h2>
<dl>
<dt>cr&egrave;me caramel</dt><dd>flan<dd>
<dt>biscuit</dt><dd>cookie</dd>
<dt>bonbons</dt><dd>candy</dd>
<dt>glace<dt><dd>ice cream</dd>
</dl>
...
</body>
</html>
```

Note that you can remove the dash (-) that separates the term from the definition as you will not need it anymore as the browser will nicely display the term and the definition of the term indented.

Images

We have seen many different types of images being used in web sites. The jpg, gif, and png files are the most common formats you will find in web sites. Sometimes you might see jpeg (the same format as jpg) and the new format webp (still not widely supported by browsers).

The jpg format is generally used for photographs, very detailed images. The gif format is generally used for cartoonish pictures. The png has been the format that is lately trying to substitute the gif format as it offers more colors and can also accept transparency (when an image has transparency it means that you can see through the image).

It’s important to understand that pictures should not be used simply to “color” a web page. They should have some purpose and be closely related to the content otherwise they will serve only as a distraction.

To include an image, you will use the <img> tag. This tag does not have a closing tag but it has some required attributes:

- **src** – the value of this attribute brings the location and the name of the picture file to be shown
- **alt** – the value of this attribute has a short description of the image being shown

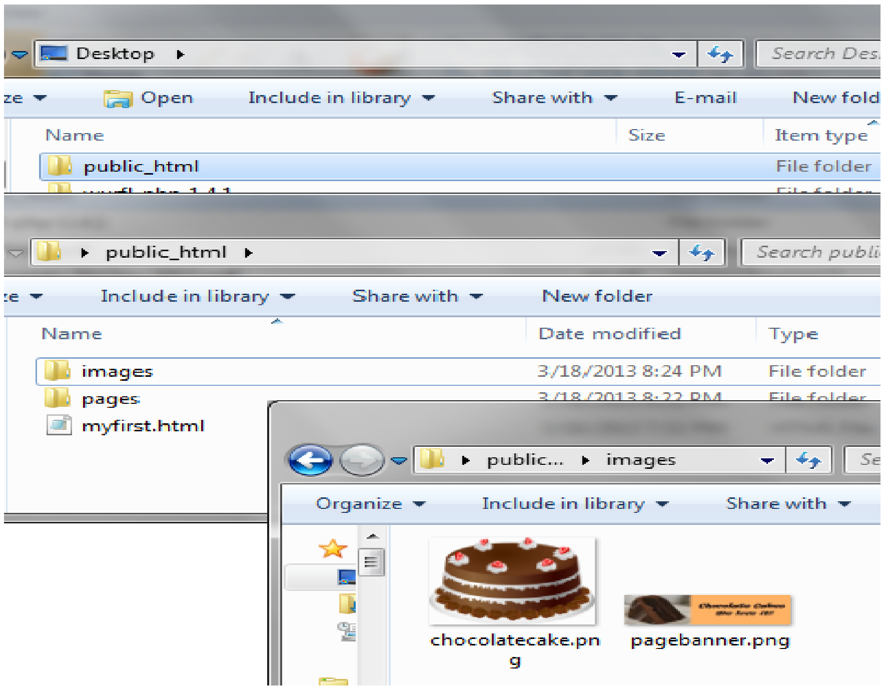
The **alt** attribute is being required since XHTML for accessibility purposes. The content (value) of this attribute will be read by screen readers generally used by people with vision disabilities. Some developers have coded the alt attribute as `alt=""` – this is not a good practice as you are just making your code be valid by the W3C validator but your code will not be accessible, meaning that people with vision disability will not know what is the message the image is bringing to the page as there is no value (content) coded for the alt attribute.

Without the **src**, no image will be shown as the browser will not be able to “guess” which image you want to show on the page. The value of this attribute gives the path to the image file and it might have the absolute or relative path to the image file.

A break for an explanation here (although the name of the html file is similar to the one we are working with, the images presented in the example shown here are not related to the images or the structure of files and folders we are using with this ebook) – the purpose of the image you see below is simply to make you understand what is the path to a file.

When we mention path to a file, we might be talking about absolute or relative path. In order to understand both concepts, you need to have in mind the structure of files and folders that exist in a system that might compose a website (remember that a website might have 1 or more pages and each of the page(s) might refer to 1 or more files such as: images, videos, documents, etc.). In a real world situation, developers will NOT put all the files inside the same folder as generally they tend to organize their files in specific folders – for example, you might have a folder called **images** where all the images will be stored; you might have another folder called **documents** where all documents (such as pdf files) will be stored; etc. Developers need to know the structure of files and folders they will be working with and that same structure of files and folders need to be maintained at the web server, otherwise developers might end with lots of “NOT FOUND” files (images not showing up on the page(s)).

What is *absolute and relative path*?



In a computer, or even in the web server, files are organized in folders and even inside a folder you may have sub folders. Every web page has a root (web root) and inside that area, a web developer may need to create different folders to categorize the files that will be used by the web site – example: like mentioned in the previous paragraph, you may have a folder called images, another called styles, another called documents, etc.

When you are editing the code of an html file, you should imagine yourself “inside” that file which may be inside a folder or right at the web root. From that “place” whenever you try to reach other files, whenever you want to present other files on your page, you will need to point at those files you want to use so the browser can “find” those files (browsers will ONLY follow your directions, they will NOT be searching randomly all over the net for your files). The HTML document will not “carry” the files you will be pointing to, for example, if you have the code, in your HTML file, shown below, it does not mean that your HTML file (your web page) will be “carrying” or “have attached” the **chocolatecake.png** file with it. The HTML code is simply giving the PATH so the browser can “find” the chocolatecake.png file to show on the browser when showing your web page.

```
<img src="images/chocolatecake.png" alt="delicious chocolate cake
with cherries on top">
```

Imagine that you have the structure shown on the image above in a web server: The web root is the **public_html** folder. When you open that folder, you can see that there are two folders: **images** and **pages** and one file **myfirst.html**.

In the example shown here, inside the images folder, there are two files: **chocolatecake.png** and **pagebanner.png**.

So, if you are editing myfirst.html, which is at the web root folder (public_html), and you want to insert the chocolatecake.png in that page, the relative path to that image will be **images/chocolatecake.png** (with that, you would be “saying” to the browser: “From where I’m coding, in order for you, Mr Browser, to find the image chocolatecake.png, you need to open the images folder first”)

On the other hand, if the image was at the same folder (location) as the file you are editing, you would simply give the name of the image file and no path is needed so the src value would simply be chocolatecake.png. Your code would be:

```
<img src="chocolatecake.png" alt="delicious chocolate cake with cherries on top">
```

The absolute path would be the entire path to find the image which, in case of a web page, would be http://domainname/images/chocolatecake.png.

Imagine that the company of this example decides to open an account in Instagram, or Pinterest, or Flickr which are websites used to store and share images, pictures. The company could decide to not upload the image files to the web server and, instead, whenever they want to show the images in their web pages, they would need to use the absolute path to point to the exact image you want to show on your web page.

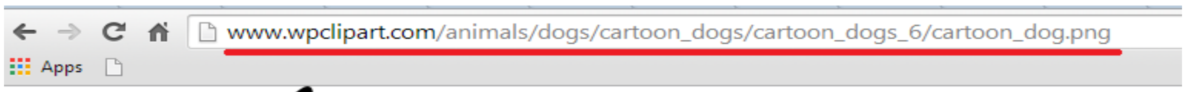
Example: If you wanted to show the image (logo) of Wikipedia website, you would write the following img tag in your HTML document:

```
<img
src="http://upload.wikimedia.org/wikipedia/commons/6/63/Wikipedia-
logo.png" alt="front page image of Wikipedia website">
```

You can test this code if you want and you will see that it really works. This procedure has been used to increase traffic between the website (server of the site) that stores your media files and your own website (server of your site) and many times it helps in the download time as those media web servers tend to be faster than normal web servers used by small businesses. One disadvantage of this procedure is if the image changes location, especially if you are pointing to an image that is not under your administration (like the example we have shown here with the Wikipedia logo – if the owner of the Wikipedia logo decides to change the file name, or the location of the file, then the code above would not work and no image, no logo would be presented on the browser). The other disadvantage is in regards to Search Engine Optimization (SEO) – although you are increasing traffic between your web page and other well-known websites (such as Instagram, Flickr, Pinterest, etc.), you are, in reality helping THOSE websites to rank better in search engines especially when people find your page via the name of the image, you are not helping YOUR website to rank better with this strategy.

ATTENTION ABOUT COPYRIGHT ISSUES!!! Another detail to remember is that whenever you are using images, or video, or audio, you need to make sure if you have the right to use those files so you do not face copyright issues.

How do you get the URL (the absolute path) of an image? Once you see the image on the browser, you can right-click on it and “copy the image URL” and then paste it right at your code or you can copy the URL of the image that is shown in the address bar when you are displaying the image. See the example below where the image of a dog is shown and the URL (absolute path) is shown in the address bar of the browser, underlined with a red line. This URL would be value of the src attribute in the img tag:


www.wpclipart.com/animals/dogs/cartoon_dogs/cartoon_dogs_6/cartoon_dog.png
Apps

The img tag to show that dog would be:

```
<img
src="http://www.wpclipart.com/animals/dogs/cartoon_dogs/cartoon_dog
s_6/cartoon_dog.png" alt="brown cartoon dog with a bone">
```

Now, let’s go back to our situation here where we will be uploading and storing the images of our web page in the same folder where we will be storing our html document. You will then code the HTML based on this situation.

So, let’s keep working with myfirst.html in your text editor (hopefully you are working with Adobe Brackets) and include, right before **Our products and prices** the following code:

```
<img src="jelly.png" alt="strawberry jelly">
```

Your code will look like:

```
     ...
<body>
```

```
...
</article>
<hr>
<article>
<h3>…</h3>
<img src="jelly.png" alt="strawberry jelly">
<h2>Our products and prices</h2>
...
</body>
</html>
```

Save your file and open it in a browser (or refresh the browser if the file was already open there) and verify that the picture is being shown. You should see the horizontal rule (line) you coded and then the picture of the jelly. If you do not see the image of the jelly, you should check those two possibilities:
a) you did not type the right code shown above;
AND/OR
b) you did not copy. to the **webroot** folder in your Desktop, the 4 files mentioned in the part of this tutorial about Organizing your Computer – which means that the jelly.png image is not inside the webroot folder.

Suppose that you would like to insert a short description of the image right below the image on your page. If you thought about using the <br> tag and then writing the text, you are right – that would be one solution to that problem and we could even code the img tag such as shown below:

```
<p><img src="jelly.png" alt="strawberry jelly"><br>Strawberry
Jelly</p>
```

Notice that I even included the img tag and the text within a paragraph. That’s the solution that web developers have used for many years and some continue to use.

HTML5 brought the <figure> tag with the <figcaption> tag to facilitate and make the code more semantically correct. The code with the img tag you typed before in myfirst.html, could be coded as:

```
<figure>
<img src="jelly.png" alt="strawberry jelly">
<figcaption>Strawberry Jelly</figcaption>
</figure>
```

The words ***Strawberry Jelly*** will be shown right below the picture. If you want the words to be shown above the picture, just place the `<figcaption>Strawberry Jelly</figcaption>` above the img tag.

Note:

Remember when we mentioned about some companies using the strategy of creating accounts in websites such as Instagram, Flickr to save their images and then using the absolute path for the src attribute of the img tag instead of uploading the image files to the web server? Well, this also applies to the figure tag. So, for example, if you have an image that you uploaded to Flickr or Instagram and that image has the absolute path of **https://farm3.staticflickr.com/2841/13385621434_a4d8c59a04_s.jpg** then, you would have your figure tag coded as:

```
<figure>
<img
src="https://farm3.staticflickr.com/2841/13385621434_a4d8c59a04_s.
jpg" alt="some image from Flickr">
<figcaption>The caption to the image is here</figcaption>
</figure>
```

Links

Links, or hyperlinks, were the drive for the creation of HTML. At that time, it was important to be able to link one research document to another and that’s how everything started.

Links are created in HTML using the anchor tag **<a> </a>**. This tag has an important attribute **href** which is used to give the relative or absolute path of the file (generally a web page) that you want to link to (go to).

Links can be found within a paragraph or isolated (topics, menu items) and you should always choose meaningful words to be used for your links and avoid words such as “click here”, or “here”, or “click” mainly when thinking about accessibility. The chosen words will be surrounded by the **<a>** and **</a>** tags.

Whenever researching about building web pages, you will read terms such as “inbound link” and “outbound link”.

- **Inbound links** – links that are targeting your page(s); links from other web sites that are coming to your site (pages). Example: ABC Company inserts a link in one of its pages to your company’s website – from the perspective of your company, this is an inbound link.
- **Outbound links** – links from your web site (pages) that are “going out” to other web sites (pages). Example: Your company web site inserts a link to ABC Company web site – from the perspective of your company, this is an outbound link.

We will create a link in our myfirst.html file. It will be an outbound link to a fake company called All Culinary Schools. You will substitute the words All Culinary Schools (at the end of the paragraph that comes before the unordered list) by <a href=”http://www.acschools.com”>All Culinary Schools</a>. Your code should look like:

```
     ...
<body>
...
<p>... such as the <a href="http://www.acschools.com">All Culinary
Schools</a>.</p>
...
</body>
</html>
```

Test your page in a browser. You will notice that when you click on the link, your page will be replaced by the page with the URL http://www.acschools.com – you might see a page but when this ebook was written, this was a fake URL which means that you might probably see nothing or the NOT FOUND message.

What if you want to open that new page in another window or tab of the browser? You can use the attribute **target** in the anchor tag with the value **_blank**. Your code would then be `<a href="http://www.acschools.com" target="_blank">All Culinary Schools</a>`.

You are now imagining the obvious: that it’s possible to create links to other pages within your web site, right? Yes, it is. You will insert a very simple menu at the top of your page. Open again myfirst.html in your text editor and include, right after the h1 closing tag, a paragraph with the words Home Mission. Your code should look like:

```
     ...
<body>
<header>
<h1>Cakes, Jelly, Desserts</h1>
<p>Home &nbsp;&nbsp; Mission</p>
```

```
</header>
...
</body>
</html>
```

Remember special characters? We have just used one of them here twice - &nbsp; - to create two extra blank spaces between the words Home and Mission. If you hit 3, 4, 5, etc. times the space key, it will NOT create the spaces in the browser. The browser will only show a single space no matter how many times you hit the space bar of your keyboard – so, if you need more spaces, use the special character &nbsp;

Save your file and take a look at your page in a browser. Up to now, of course, we do not have links, you have only the two words Home and Mission separated by 4 blank spaces (two that you typed and two from the two special characters you coded). Let's then create another page.

You will keep the same file – myfirst.html – opened in your text editor and you will substitute all the content after the paragraph with the words Home and Mission, until the closing body tag by a simple paragraph with "Our mission is simply to allow you to have an unforgettable experience after tasting any of our desserts." Your code, for this new page, will look like:

```
     ...
<body>
<header>
<h1>Cakes, Jelly, Desserts</h1>
<p>Home &nbsp;&nbsp; Mission</p>
</header>
<section>
<h2>Our Mission</h2>
<article>
<h3>We are here for you…</h3>
<p>Our mission is simply to allow you to have an unforgettable
experience after tasting any of our desserts.</p>
```

```
</article>
</section>
</body>
</html>
```

BIG REMINDER!!!

Notice that we are not presenting the code above the <body> tag but, of course, EVERY web page, EVERY HTML document will need to have the <!DOCTYPE html> tag, then you have the <html lang=”en”> tag, then you have the <head>…..</head> and inside that area (between <head> and </head>) you should have the <title> of the page, you should have the two meta tags (charset and description). Remember, every web page should begin with the following structure (of course, substituting “Title of the page” by a title that is meaningful to the page and also the “summary of the content of the page” by a sentence that summarizes the content of the page):

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Title of the page</title>
<meta charset="utf-8">
<meta name="description" content="summary of the content of the page">
</head>
```

Let’s get back to our new page that we were building!!!

YOU WILL NEED TO USE File / Save As from the menu of Adobe Brackets, otherwise, if you just use the File / Save, you will overwrite the **myfirst.html** file and WE DO NOT WANT THAT, RIGHT? So, after selecting File / Save As, you will save this new file as **mission.html** – it should be saved in the same folder as myfirst.html, meaning it should be saved inside the **webroot** folder!

Note:
Remember the <meta name=”description” content=”…”> and the <title>…</title> tags?
You now have two pages in your website and you should change the content for these two tags – it’s a good practice that helps with SEO (Search Engine Optimization). So, go ahead and change the content of between <title> and </title> tags and also the value of the content attribute of the meta tag description to be more related to the page you have just created!

Although we have now two pages for our web site, we still have not created the links to these pages using our little menu.

Open **mission.html** and substitute the word Home by `<a href="myfirst.html">Home</a>` and save this page. Open **myfirst.html** and substitute the word Mission by `<a href="mission.html">Mission</a>` and save the page. Your code for mission.html should look like (notice that here I am showing the code since the DOCTYPE tag!):

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Mission of Cakes, Jelly, Desserts</title>
<meta charset="utf-8">
<meta name="description" content="mission of cakes, jelly, desserts
where you can taste great desserts">
</head>
<body>
<header>
<h1>Cakes, Jelly, Desserts</h1>
<p><a href="myfirst.html">Home</a> &nbsp;&nbsp; Mission</p>
</header>
<section>
<h2>Our Mission</h2>
<article>
<h3>We are here for you…</h3>
```

```
<p>Our mission is simply to allow you to have an unforgettable
experience after tasting any of our desserts.</p>
</article>
</section>
</body>
</html>
```

The paragraph, after the <h1> element, in myfirst.html, will look like:

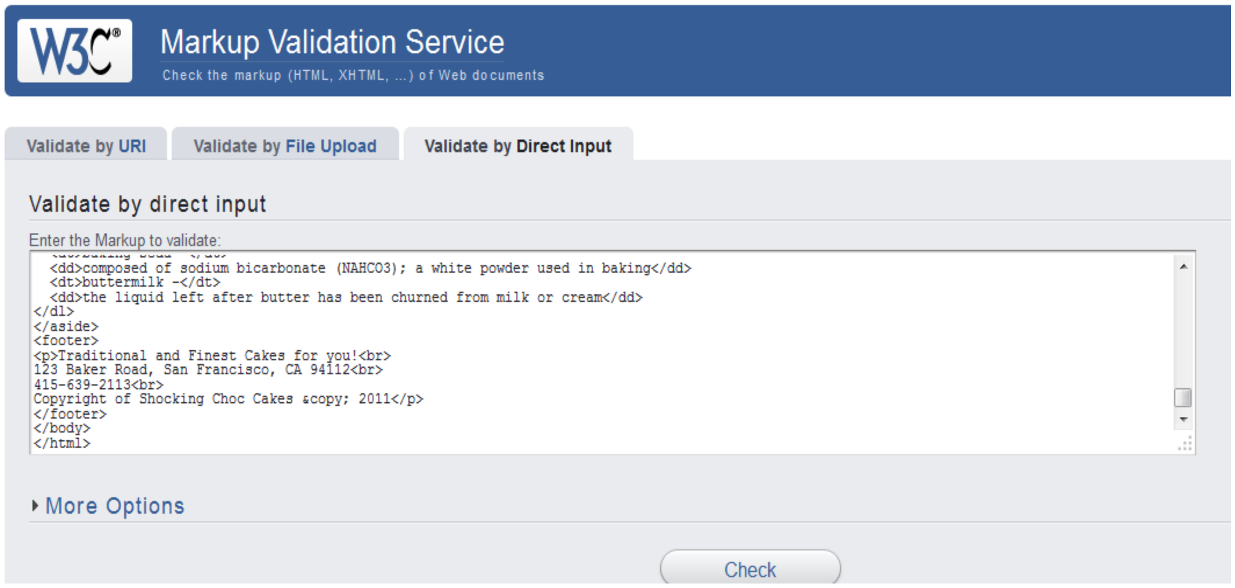
```
<p>Home &nbsp;&nbsp; <a href="mission.html">Mission</a></p>
```

Now, open myfirst.html in a browser and click on the link Mission – the mission.html page should open replacing myfirst.html page and when the Mission page shows up, you should be able to click on the Home link and get back to myfirst.html page.

Note:
Remember the difference between relative and absolute path?
This applies to links as well, meaning that if you would save mission.html inside another folder, such as pages, then, the right value for the href attribute in myfirst.html would be **pages/mission.html** and, in this case, to have the link to go back to the home page (myfirst.html), in the mission page, would be **../myfirst.html**.
The **..** means that the link (the page) is located one level up than the page with the link code.

Validating the HTML code

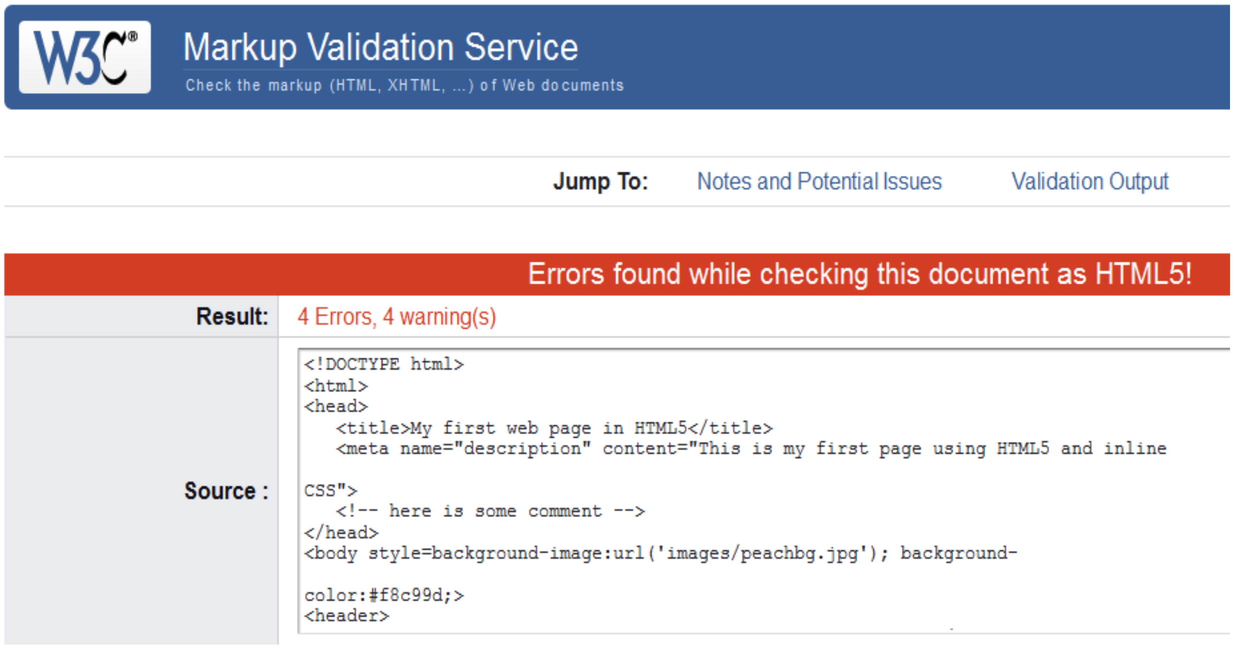
It's always good to make sure your HTML code is valid. To validate the code, you should open the W3C Validator (http://validator.w3.org). You can validate the HTML code after loading the web page to the web server and using the URL to validate, or by uploading the html file to the validator, or directly inputting the code into the validator. Take a look at the validator page below showing the "**Validate By Direct Input**" window opened and the HTML code to be validated, already inserted in the window.



```
  <dd>composed of sodium bicarbonate (NAHCO3); a white powder used in baking</dd>
  <dt>buttermilk -</dt>
  <dd>the liquid left after butter has been churned from milk or cream</dd>
</dl>
</aside>
<footer>
<p>Traditional and Finest Cakes for you!<br>
123 Baker Road, San Francisco, CA 94112<br>
415-639-2113<br>
Copyright of Shocking Choc Cakes &copy; 2011</p>
</footer>
</body>
</html>
```

Once the code is all inside the box, you can click on the Check button (also shown in the picture above) and the result will be shown in a page that can look like these shown here:

a) This picture shows the usual screen when the HTML code is considered invalid (some errors are found) – notice the red bar


Markup Validation Service

Check the markup (HTML, XHTML, ...) of Web documents

Jump To: Notes and Potential Issues Validation Output

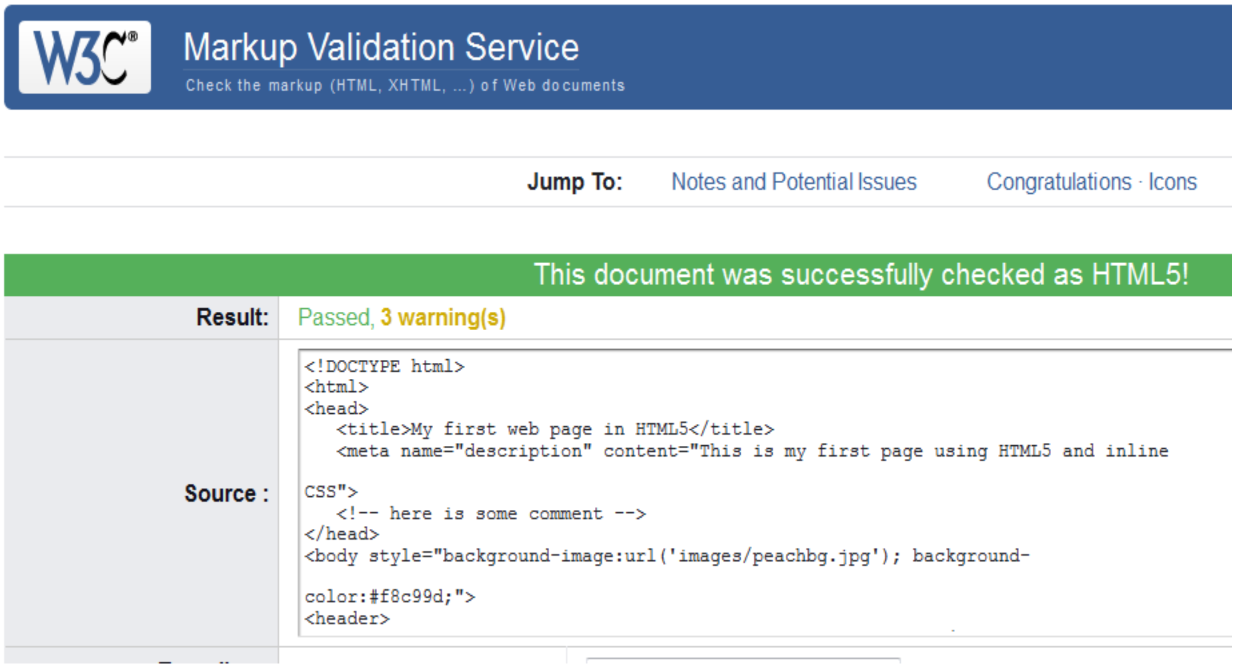
Errors found while checking this document as HTML5!

Result: 4 Errors, 4 warning(s)

Source :

```
<!DOCTYPE html>
<html>
<head>
   <title>My first web page in HTML5</title>
   <meta name="description" content="This is my first page using HTML5 and inline

CSS">
   <!-- here is some comment -->
</head>
<body style=background-image:url('images/peachbg.jpg'); background-

color:#f8c99d;>
<header>
```

b) This picture shows the usual screen when the HTML code is considered valid (even with some possible warnings) – notice the green bar


Markup Validation Service

Check the markup (HTML, XHTML, ...) of Web documents

Jump To: Notes and Potential Issues Congratulations · Icons

This document was successfully checked as HTML5!

Result: Passed, 3 warning(s)

Source :

```
<!DOCTYPE html>
<html>
<head>
   <title>My first web page in HTML5</title>
   <meta name="description" content="This is my first page using HTML5 and inline

CSS">
   <!-- here is some comment -->
</head>
<body style="background-image:url('images/peachbg.jpg'); background-

color:#f8c99d;">
<header>
```

When you get the error screen (invalid HTML code), you will need to scroll down the page to read the errors that were

found – the narrative will show the line number where the error was found.
Validating your code is also a great exercise to understand the narrative of the error found and be able to spot the real problem in your code – as you evolve in experimenting the W3C validator, you will notice that many times, when you fix a small error, you will cause many of the listed errors to disappear.

Note:
Just because a web page is shown in the browser, it does not mean that it carries a valid HTML code. A valid HTML code is the one that follows the standards set by the World Wide Web Consortium (W3C). It's important to have a valid HTML for SEO (Search Engine Optimization) and also to "easy" the browser work when parsing (interpreting) your HTML code.

Special Note About Tags Used

Basics about block or inline tags

We have used many different tags in the exercises we have done so far but there is one important point to be discussed which is the difference between block or inline tags.
A block tag is the one that when parsed by the browser (presented on the browser) will be presented as a block, as an independent piece of block. For example, tags such as h1, p, section, article, div are considered block tags which

means that by default, when I write two paragraphs in my HTML document, even if they are very short paragraphs, they will NEVER be presented side by side on the browser (in the same line) as they will be presented in two separated blocks, one below the other.
The img tag, on the other hand, is a clear example of an inline tag – when we code the img tag, it will not be presented in a different line/block, it will be presented INLINE with the content, in the same line of the rest of the content.
This is a very important notion to have when later on, in more advanced "adventures" with HTML & CSS, you might want to modify the default behavior of a tag using what is called the display property of a the element (the tag).
Based on this explanation, and based on what you have been practicing with the previous exercises, how would you then classify the following tags:
<li>
<table>
I hope you have answered that these two are also block tags, right?

Introduction to CSS

What exactly is CSS?

CSS is not a programming language either! It's short for Cascading Style Sheet.
When tags such as font (<font>) and color attributes were added to HTML, the final HTML document started to look really confusing and sometimes more tags or attributes were found instead of content. This made it more difficult to maintain web pages.
CSS started in 1994 as the web was starting to be used to publish documents – something was missing as it was not possible to format or to layout those documents. Also, the idea of separating the formatting from the content was not new but the initial proposal was presented at the Web Conference in Chicago in 1994.
In 1995 the World Wide Web Consortium (W3C) became operational and by 1997 the CSS had its own working group inside the W3C to work on the features not addressed by the first release of CSS (CSS1). In 1998 the CSS2 became a recommendation and work started to happen for CSS3 in 1999. By the middle of 2012, there were over fifty CSS3 modules published.
So, in summary, web developers use CSS to style/format their web pages.

CSS Syntax

CSS has two main parts: a selector and one or more declarations and each declaration is made of a property and a value.
Example:
h1 {font-size: 3em; color: red;} - this example shows the selector **h1** with two declarations (for **font-size: 3em** and **color: red**). The first declaration uses the property **font-**

size with the value of **3em** and the second declaration uses the property **color** with the value **red**.
A CSS declaration should always end with a semicolon (;) as you see in the example.
This syntax is normally found in embedded or external CSS. But what exactly is this?

Embedded CSS, External CSS, Inline CSS

You can code the CSS using three different styles: inline, embedded, external.

External CSS – the CSS code is included in an external file that will always have the extension .css. When you open a .css file (in a simple plain text editor), you will see the CSS written using the style shown in the example above. The external file is linked to an HTML document using the <link> tag in the head section of the HTML document.

Embedded CSS – also known as document-level CSS. The CSS code is written within the head section of an HTML document and all the CSS will be surrounded by the opening and closing <style> tag (`<style>` … `</style>`) – developers used to code the opening <style> tag as `<style type="text/css">` but currently you would see a warning from the validator to mention that you do not need to code the attribute ***type*** anymore – that's what you see in the code below:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Page with CSS</title>
<style>
h1 {font-size: 3em; color:red;}
h2 {text-align:center;}
p {first-style: italic;}
</style>
</head>
<body>
…
```

```
</body>
</html>
```

The bold red part in the example above shows an embedded CSS.

Inline CSS – this code will come inside the tag with the **style** attribute. Once the style attribute is added, the declarations will be written within double quotes. See the example here:

```
<h1 style="font-size: 3em; color: red;">My heading 1</h1>
```

The inline CSS can NEVER be shown in the closing tag. Observe that the properties and values, used in the inline CSS will be the same ones used in an embedded or external CSS; the main difference is the use of double quotes instead of curly brackets to surround the declarations and the existence of the style attribute.

In this ebook, we will focus in the inline CSS and we will continue to use the files that we used to learn HTML.

Too many properties… Too many values…

As you start using CSS, you will notice that there are many different properties and values that you can use to format the content of your page. Will we memorize all those properties and their respective values? Of course not! It's recommended that you use a reference and as you practice, you will certainly know some of those properties and values "by heart".

Recommended sites:

- http://www.htmldog.com/reference/cssproperties/
- https://developer.mozilla.org/en-US/docs/CSS/CSS_Properties_Reference
- http://www.htmlhelp.com/reference/css/properties.html

- http://meiert.com/en/indices/css-properties/

There are other web sites with good CSS references and here are just some examples.
You should choose one web site so whenever you need to use it as a reference to find the most appropriate property and value to use for your formatting purposes.

Going back to your html document...

I think that now you are ready to start using some inline CSS to enhance the web page we have been building since the beginning of this e-book.
We will reopen **myfirst.html** file and we will start by adding a background image and a background color to our whole web page. We will use the website http://www.sfsu.edu/~jtolson/textures/textures.htm to choose one light color texture for our background image. I have chosen and downloaded the file which I named **peach.jpg**. You can use this file for our exercise and then choose another one to test.

Note:
There are many different websites from where you can download images to use for background of your pages but you need to be careful to not infringe copyright law.

Open **myfirst.html** and then add the style attribute to the body tag (code shown below) – pay attention to the **style** attribute that was added in the opening body tag.

```
     ...
<body style="background-image:url('peach.jpg')">
<header>
<h1>Cakes, Jelly, Desserts</h1>
...
</body>
```

```
</html>
```

Note:
Remember that you cannot add attributes to closing tags, meaning, you would not add the style attribute to the </body> tag.
Observe that style is an attribute – you type style followed by the equal sign (=). Then, the value of any attribute in HTML is surrounded by double quotes (“ “). The value of the style attribute will be the CSS property and the CSS value of the property to achieve the desired format. In our case, the CSS property we need to use is the background-image which is followed, according to the CSS syntax, by colon (:) and then the value for the CSS property (in our case, it is url(‘peach.jpg’)) – that’s the way your insert the value for background-image. When finished the property and value of the property, you type the semicolon (;).
We could have other CSS properties and their values inside the same style attribute and they would all be separated by the semicolon (;) AND a space. We will see how to do this one next in that same body tag when we will add the background-color property.
The name of the file that contains the background-image we want is peach.jp but remember that we need to give the full path where that image will be found and if we decide to create a folder called **images** to store all our images files, we will need to include this folder in the path as `background-image:url('images/peach.jpg');` in the code.

Now, we will include, in this same style attribute, the background-color as seen below:

```
     ...
<body      style="background-image:url('peach.jpg');      background-
color:#f8c99d;">
<header>
<h1>Cakes, Jelly, Desserts</h1>
...
</body>
</html>
```

Note:
You have to be careful when using background images as it will add to the download time of your page especially if you choose images that are too heavy in terms of bytes.
Another detail to pay attention to is the fact that your page must be readable so you need to be extra careful when choosing background images or background color and the color of the font you will use to build your pages. You want the users to read the content, right?
It's good to define a background-color, when defining a background-image, preferably having the background-color that resembles the dominant color of the background-image. In case your image takes too long to download or it gets deleted inadvertently, it's always good to let the users with the same "look-and-feel" you had offered before when using the background-image. If you choose a background image that you do not know the name or the hexadecimal representation of the color, you can visit the website http://bighugelabs.com/colors.php where you will be able to upload your image and hit the Create button to have a good choice of colors (in hexadecimal value) that are very close to the image you have chosen to be the background of your page. That's the reason you see **background-color:#f8c99d;** coded in our example.

Save your myfirst.html file and open it in the browser. It works, right? If it does not work, remember to look at the CSS code you typed and make sure that the property is correctly spelled as the browser will not understand if you type, for example, **backgound-color** (<u>MISSING THE r to make the word g**R**ound</u>)! Check also if you entered a space after the semicolon (;) of the first property before you typed the second property (background-color).

There are more CSS properties that can be used together with the background-image such as:

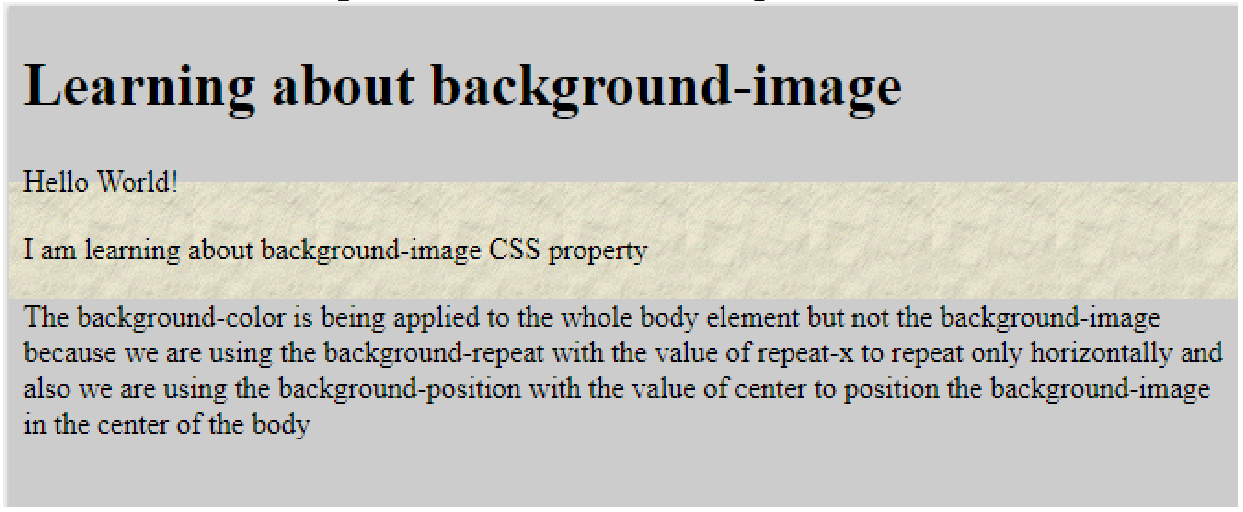
- **background-repeat** – when this property is not used, the image of the background-image will repeat ALL over the element to which the background-image property was applied. You can use the value repeat-x and then your background-image will repeat only in a horizontal line pattern. If you use the repeat-y, the background-image will repeat only in a vertical line pattern. If you use no-repeat, it will only "print" the background-image one time in the top left part of the element to which you applied the image
- **background-position** – this property will only make sense to use if you also use the background-repeat with one of the values explained above as you can set up the position of the background-image within the element to which it was applied

There are other CSS properties related to the background-image but these two would be enough for you to play around with it. See some examples below:

```
<body     style="background-image:url('paper.gif');     background-
color:#CCCCCC;     background-repeat:     repeat-x;     background-
position:center;">
<h1>Learning about background-image</h1>
```

```
<p>Hello World!</p>
<p>I am learning about background-image CSS property</p>
<p>The background-color is being applied to the whole body element
but not the background-image because we are using the background-
repeat with the value of repeat-x to repeat only horizontally and
also we are using the background-position with the value of center
to position the background-image in the center of the body</p>
```

This code would produce the following result in the browser:



Learning about background-image

Hello World!

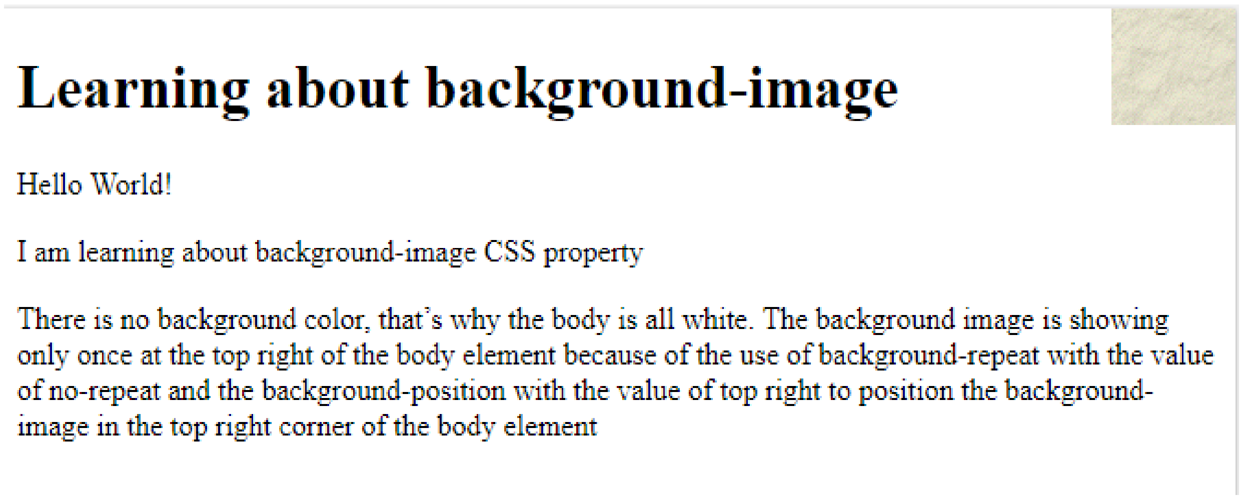
I am learning about background-image CSS property

The background-color is being applied to the whole body element but not the background-image because we are using the background-repeat with the value of repeat-x to repeat only horizontally and also we are using the background-position with the value of center to position the background-image in the center of the body

Another example:

```
<body      style="background-image:url('paper.gif');      background-
color:#CCCCCC; background-repeat: no-repeat; background-position:top
right;">
<h1>Learning about background-image</h1>
<p>Hello World!</p>
<p>I am learning about background-image CSS property</p>
<p>The background-color is being applied to the whole body element
but not the background-image because we are using the background-
repeat with the value of repeat-x to repeat only horizontally and
also we are using the background-position with the value of center
to position the background-image in the center of the body</p>
```

The code above would produce this result in the browser:


Learning about background-image

Hello World!

I am learning about background-image CSS property

There is no background color, that's why the body is all white. The background image is showing only once at the top right of the body element because of the use of background-repeat with the value of no-repeat and the background-position with the value of top right to position the background-image in the top right corner of the body element

ATTENTION!!! SOME TIPS ABOUT BACKGROUND-IMAGE!!!

When I use the background-image and the background-repeat with the value of no-repeat, the default position of the background image will be on the top left of the element that you are applying the background image to. The same way, if you apply the background-repeat with the value repeat-x, the default position of the background image will be the top of the element that the background-image is being applied to.

Working with colors

Be careful when working with colors to build your pages! There are some people that are color-blind and you pages may become unreadable depending on the choices of colors you make. Also, colors have some meanings in some cultures and as a web developer and designer you need to pay attention to this too – for example: seniors generally do not like to face a website with dark background colors while teenagers do; kids like colorful websites while adults don't; etc.

Colors can be coded by its browser name – example: red, blue, cyan, green, black, white, etc. But you need to know the name that is recognized by the browsers, meaning, you might look at a color and say: “It’s violet” but then, if you write “background-color:violet;” the browser might not show to you the color you had in mind.

Colors can be represented by their hexadecimal values – 6 characters that uniquely represent each color. You do not want to memorize all the possible combinations, do you? Of course not! So, you can use websites such as http://www.colorcombos.com/ where you will find different popular colors that you can click on and then you will be offered different combos and each color combo will bring a variety of 4-6 colors with their respective hexadecimal values. When you use the hexadecimal value of a color, you should write the pound sign (**#**) in front of the number. That’s what you see in the CSS code we used for background-color (**background-color:#f8c99d;**).

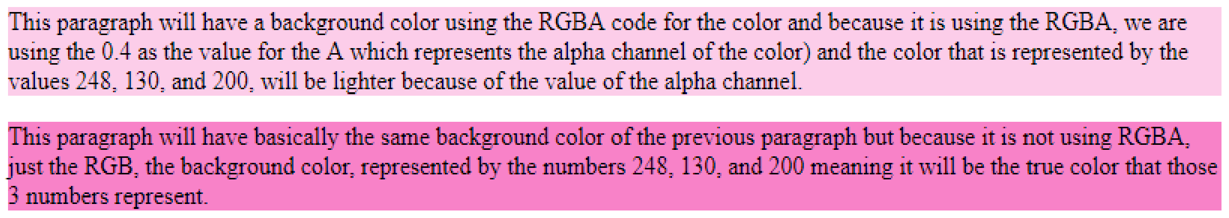
Colors can also be represented by their RGB (Red, Blue and Green) combination and you would see the code written as **background-color:rgb(a,b,c);** where a, b, and c are values that can go from 0 to 255 and they mean the intensity of each of the colors – red, blue, and green – that combined will form another color. If you do not know the RGB value of the color and you have the hexadecimal value, you can use this great website - http://www.javascripter.net/faq/hextorgb.htm - to convert your hexadecimal color value to RGB. In the case of the code presented for the background-color that we used the hexadecimal value of **f8cc99d**, the corresponding RGB value is **248**, **201**, and **9**. So the code would be **background-color:rgb(248, 201, 9);**.

CSS3 brought the RGBA (Red, Blue, Green and Alpha Channel) combination and the A (Alpha Channel) would allow you to work with the different values for opacity of the color. The value of A can vary from 0 to 1 such as: 0.2, or

0.3, or 0.6, etc. the closer to 0 (zero) closer to transparent the color will be and, of course, the closer to 1 (one) closer to opaque the color will be. So, imagine the code below:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Background Color</title>
<meta charset="utf-8">
<meta name="description" content="two paragraphs with background color using RGB and RGBA">
</head>
<body>
<p style="background-color: rgba(248, 130, 200, 0.4);">This paragraph will have a background color using the RGBA code for the color and because it is using the RGBA, we are using the 0.4 as the value for the A which represents the alpha channel of the color) and the color that is represented by the values 248, 130, and 200, will be lighter because of the value of the alpha channel.</p>
<p style="background-color: rgb(248, 130, 200);">This paragraph will have basically the same background color of the previous paragraph but because it is not using RGBA, just the RGB, the background color, represented by the numbers 248, 130, and 200 meaning it will be the true color that those 3 numbers represent.</p>
</body>
</html>
```

This code would produce the following (note that the two paragraphs have the same RGB combination but the first paragraph is lighter because it has 0.4 for the A (alpha channel = opacity) value:



This paragraph will have a background color using the RGBA code for the color and because it is using the RGBA, we are using the 0.4 as the value for the A which represents the alpha channel of the color) and the color that is represented by the values 248, 130, and 200, will be lighter because of the value of the alpha channel.

This paragraph will have basically the same background color of the previous paragraph but because it is not using RGBA, just the RGB, the background color, represented by the numbers 248, 130, and 200 meaning it will be the true color that those 3 numbers represent.

Alignment of text

In HTML5, the <center> tag is considered obsolete and the align attribute was considered deprecated not long ago but now is also considered obsolete. Whenever some code is considered deprecated, the validator might show a warning and it will be good if you then change the code with the suggestion given by the validator.

For text alignment, the correct property is **text-align**.

Let’s now centralize the heading 1 (h1) you find inside the header tag (<header>…</header>) of our myfirst.html page. Again, using the style attribute as we are using inline CSS. We will also align to the right the paragraph (<p>…</p>) you see inside the header. Look at the complete code below with these two inline CSS inserted shown here in bold:

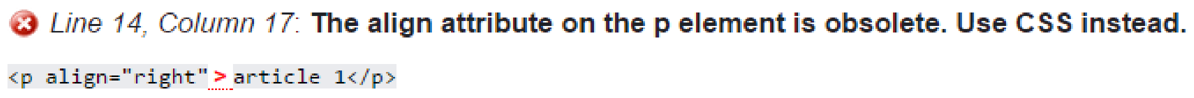
```
    ...
<body     style="background-image:url('peach.jpg');     background-color:#f8c99d;">
<header>
<h1 style="text-align:center;">Cakes, Jelly, Desserts</h1>
<p      style="text-align:right;">Home      &nbsp;&nbsp;      <a href="mission.html">Mission</a></p>
</header>
...
</body>
</html>
```

Note:
You will notice that there is another value for text-align property which is left. The default alignment of the text is left. So, if you do not want to change that default, you do not need to use CSS to specifically say **text-align:left;** as the browser will do it by default.
So, you might ask, when would I need to use this tough? Well, imagine that I had used the style attribute in the header tag as **<header style=”text-align:center;”>** -

with this code, any text (headings, paragraphs, etc.), inside the header (between the <header> and </header> tags) would be centralized (aligned to the center of the line) and then, in that case, if I wanted a particular block of text aligned to the left I would need to clearly "say" that to the browser by using **text-align:left;**. **REMEMBER THE INHERITANCE THAT CSS CARRIES!**

Save your page and show it in the browser. Better, right? You now have a background texture, the h1 aligned to the center (width of the browser window), the menu aligned to the right, etc.

Note:
In previous HTML versions, it was common to do the text alignment using the align attribute inside the tag such as:
`<p align="right">Paragraph aligned to the right side</p>`
`<h1 align="center">Heading 1 aligned to the center</h1>`
In current HTML5 standard, this procedure is considered obsolete and the validator will assign as an error such as shown in the picture below – note the red circle shown at the beginning of the note which is different than a warning that shows a yellow triangle.



Line 14, Column 17: **The align attribute on the p element is obsolete. Use CSS instead.**

```
<p align="right">article 1</p>
```

Changing the color of font

Let’s proceed with our changes and use the **color** CSS property. This property is used to change the color of the font. We will change the color of the font of the heading 1 (<h1>) and also the font color of the words *fruits and nuts with some honey* in the paragraph that starts by “*Sweets were appreciated...*”. Here, again, you will be able to use the name of the color, or the hexadecimal value, or the RGB value. Look at the code below showing the property inserted in bold:

```
     ...
<body      style="background-image:url('peach.jpg');      background-color:#f8c99d;">
<header>
<h1    style="text-align:center;    color:brown;">Cakes,    Jelly, Desserts</h1>
<p      style="text-align:right;">Home      &nbsp;&nbsp;      <a href="mission.html">Mission</a></p>
</header>
...
<p>Sweets were appreciated...<span style="color:#ff0000;">fruits and nuts with some honey</span>. But the name dessert......and sugar was manufactured.</p>
...
</body>
</html>
```

Wait a minute! Have you paid close attention to the code? Have you noticed any new different tag that was inserted in the paragraph starting by “**Sweets were appreciated...**”?

In order to have the color of the font changed to only a part of the paragraph, we had to use the **span** tag (<span>...</span>). The **span** tag will not interfere in the flow of the document, meaning that this tag is an inline tag and it is used every time you want to change the formatting of only a part of your text within a paragraph, or a heading, or a cell of a table, etc. This tag creates an inline separated section that you can then use the style attribute with different CSS properties in order to change the format of what is surrounded by the opening and closing **span** tags.

Note:
The main difference between the **span** and the **div** tags is that the **div** is a block tag, meaning that you will have a line break whenever you insert a **div** tag. In our example, we would not be able to use the **div** tag exactly because **we** did not want to break the flow of the paragraph.
See? Now you have just learned about another inline tag – the span tag!

You will continue styling your page!

We will now italicize the words Home and Mission and we will bold and have a red color to the word "sugar" in the paragraph that starts by "*During the old times...*". Here are some tips for you:

1. Bold can be achieved by using the CSS property **font-weight** with the value of **bold** (font-weight:bold;) or you can surround the content you want to bold using the **strong** tag (<strong>...</strong>).
2. Italicize can be achieved by using the CSS property **font-style** with the value **italic** (font-style:italic;) or you can surround the content you want to italicize using the **em** tag (<em>...</em>).

So, based on these tips, go ahead and insert the code in your myfirst.html file. Try to insert the code yourself before you look at the code below that shows to you the answer with the two different ways of coding the bold format.

THIS SPACE HERE IS TO GIVE YOU TIME TO THINK ABOUT WHAT YOU HAVE JUST READ.
OPEN THE FILE AND MAKE THE NECESSARY CHANGES!
WEB DEVELOPMENT EXPERTISE IS ACHIEVED WITH PRACTICE!
… … … … … …

In the code below the **font-style:italic;** is being used to italicize Home and Mission and the **<strong>…</strong>** tags being used to bold the word sugar

```
     ...
<body style="background-image:url('peach.jpg'); background-color:#f8c99d;">
<header>
<h1 style="text-align:center; color:brown;">Cakes, Jelly, Desserts</h1>
<p style="text-align:right; font-style:italic;">Home &nbsp;&nbsp; <a href="mission.html">Mission</a></p>
</header>
...
<p>During the old times <strong><span style="color:#ff0000;">sugar</span></strong> was so expensive......
</p>
...
</body>
</html>
```

The same result could be achieved with the code below that is using the **<em>…</em>** tags to italicize Home Mission and the **font-weight:bold;** to have the word sugar in bold:

```
     ...
<body style="background-image:url('peach.jpg'); background-color:#f8c99d;">
<header>
<h1 style="text-align:center; color:brown;">Cakes, Jelly, Desserts</h1>
<p style="text-align:right;"><em>Home &nbsp;&nbsp; <a href="mission.html">Mission</a></em></p>
</header>
...
<p>During the old times <span style="color:#ff0000; font-weight:bold;">sugar</span> was so expensive......</p>
...
</body>
</html>
```

Note:

You will still find developers using the **b** tag (<b>…</b>) to bold text. You should avoid using this tag as since HTML5 it is not considered, for semantic purposes, as good as the **strong** tag (<strong>…</strong>) or the CSS property **font-weight:bold;**.

Notice in the first solution presented that the **strong** tag came before the **span** tag although the order here is not so important but it's important to notice that the closing tags will be shown in the opposite order of the opening tags. We then have **<strong><span….>…</span></strong>**, or we could have **<span…><strong>…</strong></span>**. These would both get the same final result but it’s important to notice the right standard of coding HTML and nest the tags correctly.

Borders

The last CSS property we will learn is the one used to draw a border line around an element.

You can draw borders around the whole element, or only on the top, or on the bottom, or on the left, or on the right of the element.

Remember that when you read element, we might be talking about a paragraph, a heading, an image, a table, etc. You can draw a line around any of these elements.

The properties related to border are: **border-width**, **border-color**, **border-style**. For each of these properties, you can specify the location of the border such as **border-left-width: 2px;** will apply a 2px-width border to the left side of the element; the CSS rule **border-top-style: solid;** will apply a solid-style border to the top of the element, and so on.

You can code each of these properties separately or you can code all of those together as **border-right: style color width;** to apply a certain border style, border color, and border width to the right side of the element. The order of the values is not important, meaning you can write the values for color, style, and width in any order.

You can also write the CSS rule as **border: width style color;** and in this case, the border would be drawn around the element.

Let’s start with a simple example using your **myfirst.html** file.

Open the file and insert the following CSS rule **border-left-width:5px; border-left-style:solid; border-left-**

color:red; for the heading 1 (<h1>) tag continuing the style that was already there. The code would look like:

```
     ...
<body     style="background-image:url('peach.jpg');     background-
color:#f8c99d;">
<header>
<h1  style="text-align:center;  color:brown;  border-left-width:5px;
border-left-style:solid;    border-left-color:red;">Cakes,    Jelly,
Desserts</h1>
<p style="text-align:right; font-style:italic;">Home &nbsp;&nbsp; <a
href="mission.html">Mission</a></p>
</header>
...
...
</body>
</html>
```

Note:

Open myfirst.html in the browser and you might ask: why is the border so far away from the text? Remember that the heading tag is a block tag and the browser then separates the whole available line being that the default width, for the content. If you resize the window of the browser, you will notice that the border will get closer to the text that is centralized.

If you do not want the border to be drawn over the whole line, you can set up the width of the block element – in that case, the h1 element such as:

```
<h1 style="text-align:center; color: brown; border-left-width:5px;
border-left-style:solid; border-left-color:red; width:50%;">…</h1>
```

The width:50%; will make the block element (h1 element) be 50% of the width of the browser window.

How would you draw the same 5px-red-solid border on the other three sides of the element? As previously mentioned, you can **use border-right: color width style; border-top: width style color; border-bottom: style width color;** or you can simply substitute the border-left by **border: red 5px solid;**. So, let’s change the code in myfirst.html to draw a border all around the h1 element and your code will then look like:

```
     ...
<body     style="background-image:url('peach.jpg');     background-color:#f8c99d;">
<header>
<h1  style="text-align:center;  color:brown; border:  5px  solid red;">Cakes, Jelly, Desserts</h1>
<p style="text-align:right; font-style:italic;">Home &nbsp;&nbsp; <a href="mission.html">Mission</a></p>
</header>
...
...
</body>
</html>
```

The order of the settings for the border (type of border, width of the border, color of the border) is not important but notice that there is only a blank space between each of the settings value.
You will now remove the border from the heading 1 and insert it in the image (<img>) tag to draw a border around the image (the jelly.png image).

THIS SPACE HERE IS TO GIVE YOU TIME TO THINK ABOUT WHAT YOU HAVE JUST READ.

OPEN THE FILE AND MAKE THE NECESSARY CHANGES! WEB DEVELOPMENT EXPERTISE IS ACHIEVED WITH PRACTICE!
… … … … … …

Your code should look like:

```
     ...
<body     style="background-image:url('peach.jpg');     background-color:#f8c99d;”>
<header>
<h1    style="text-align:center;    color:brown;">Cakes,    Jelly, Desserts</h1>
<p style="text-align:right; font-style:italic;">Home &nbsp;&nbsp; <a href="mission.html">Mission</a></p>
</header>
...
<p>During the old times ...... <span style="color:##FF0000; font-weight:bold;">sugar</span>...
...
<hr>
<article>
<img src="jelly.png" alt="strawberry jelly" style="border:5px solid red;">
<h2>Our products and prices</h2>
...
</body>
</html>
```

And the image would then show on the browser with a red-5px-solid border.

Now, you have a very good basis to continue your studies of HTML and CSS and you will find out that both have evolved a lot and the more you learn, the more you will be able to build great web sites.